FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Learning and Indexing of Texture-Based Image Descriptors in Medical Imaging Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Johannes Hofmanninger
Matrikelnummer 0725497

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: a.o.Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Ass.Prof. Dipl.-Ing. Dr. Georg Langs (CIR Lab, Medical University of Vienna)

Wien, TT.MM.JJJJ

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Learning and Indexing of Texture-Based Image Descriptors in Medical Imaging Data

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Medical Computer Science

by

Johannes Hofmanninger
Registration Number 0725497

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: a.o.Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: Ass.Prof. Dipl.-Ing. Dr. Georg Langs (CIR Lab, Medical University of Vienna)

Vienna, TT.MM.JJJJ

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Johannes Hofmanninger
Aistersheim 93, 4676 Aistersheim

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Danksagung

Ich möchte mich an dieser Stelle bei all denjenigen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben. Ganz besonders gilt dieser Dank Dr. Georg Langs vom CIR-Lab an der Medizinischen Universität Wien, der diese Arbeit betreut hat. Nur seine zahlreichen Anregungen, Ideen und Ratschläge haben es ermöglicht die Arbeit in dieser Form fertig zu stellen. Nicht zuletzt war auch er es, der durch seine ausgezeichneten Vorlesungen mein Interesse an der medizinischen Bildverarbeitung geweckt hat.

Es ist mir auch ein besonderes Anliegen mich bei all meinen Kollegen und Freunden am CIR-Lab für die angenehme, produktive und kollegiale Zusammenarbeit zu bedanken. Vor allem jedoch bei Thomas Schlegl, René Donner und Markus Holzer für Ihre Beiträge zu dieser Arbeit. Ganz herzlich möchte ich mich auch bei Dr. Robert Sablatnig vom Computer Vision Lab an der Technischen Universität Wien für die offizielle Betreuung und die hilfreichen Kommentare bedanken.

Zuletzt möchte ich mich bei meiner Familie und ganz besonders meinen Eltern bedanken, die mich während meiner gesamten Studienzeit weit über das Maß der Selbstverständlichkeit hinaus unterstützt haben.

Abstract

Medical imaging is used for diagnosis and monitoring in the daily routine of hospitals. Up to thousands of images are recorded every day in just a single hospital. A way to increase the accessibility of such massive image databases is Content Based Image Retrieval (CBIR). Rather than text based search engines, CBIR uses actual visual information to find visual similar images for a query image. *Visual similarity* is a subjective terminology that is bound to a certain context. Thus, no standard design or methodology to solve a CBIR problem exists.

The present work focuses on the development and evaluation of specialized CBIR components for three dimensional medical images. One part of the work deals with the development of a visual descriptor (a numerical representation to describe visual features) that allows to compare the similarity of images. Typically, such a descriptor is visual data driven. Medical images may be provided with textual information in the form of meta data or radiology reports. This information conveys semantic information as it indicates radiological relevant visual appearances. We propose a way to link visual features to such textual information.

Besides the descriptors, a CBIR system needs an indexing method that enables fast nearest neighbor search functionality. Approximate nearest neighbor indexing methods are used to search the database of descriptor vectors.

Real world data, recorded in the daily routine of a hospital is used to evaluate the components developed. Four anomalies that are visible in the lung are defined as information need. The results show, that semantic information improves the ranking of relevant images for three of the four anomalies. Effects of the approximate nearest neighbor search on the ranking quality are studied. Results show, that the use of approximate search methods can reduce the memory consumption by 98% without decreasing the retrieval quality considerably.

Kurzfassung

Medizinische Bildgebungsverfahren gehören zu grundlegenden Instrumenten der klinischen Routine, die zur Erkennung und Überwachung von Krankheiten eingesetzt werden. Durch die Archivierung dieser Bilder entstehen große Datenbanken. Content Based Image Retrieval (CBIR) bietet eine Möglichkeit die Nutzbarkeit dieser Daten zu erhöhen. CBIR bezeichnet die Suche auf Basis von Bildinhalten bei der die Anfrage durch Bildinformation gestellt wird. Da der Ähnlichkeitsbegriff anwendungsspezifisch ist, werden auf Daten und Anwendungsziele angepasste Methoden benötigt.

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung und Evaluierung von Methoden, die für die Suche von dreidimensionalen medizinischen Bilddaten geeignet sind. Eine besondere Eigenschaft klinischer Daten ist die Verfügbarkeit von radiologischen Befunden, welche relevante visuelle Erscheinungen im Bild beschreiben können. Der erste Teil der Arbeit beschreibt die Entwicklung eines visuellen Deskriptors, der es ermöglicht, radiologisch relevante Unterschiede von Bildtexturen zu beschreiben. Während solche Deskriptoren in der Regel auf Basis rein visueller Information definiert werden, beschreiben wir eine Möglichkeit semantische Information in Form von Schlagwörtern mit visuellen Merkmalen zu verbinden.

Der zweite Teil der Arbeit identifiziert und vergleicht geeignete Indizierungsmethoden um eine große Anzahl an Deskriptoren schnell vergleichen zu können.

Der entwickelte Deskriptor als auch die Indizierungsmethoden werden mit Hilfe eines realen Datensatzes eines Krankenhauses evaluiert. Vier Anomalien werden als Beispiele für relevante Bildveränderungen definiert. Die Ergebnisse zeigen eine verbesserte Reihung relevanter Bilder für drei der vier Anomalien wenn semantische Information bei der Erstellung des Deskriptors verwendet wurde. Des weiteren werden die Effekte der approximativen Suchmethoden auf die Reihung relevanter Bilder Analysiert. Wir Zeigen, dass neben einer Erhöhung der Suchgeschwindigkeit auch der Speicherverbrauch um 98% gesenkt werden konnte ohne die Suchqualität deutlich zu mindern.

Contents

Acronyms

ADC	Asymmetric Distance Computation
ANN	Aproximate Nearest Neighbor
AP	Average Precision
APL	Active Page List
AUC	Area Under the ROC curve
BoVW	Bag of Visual Words
CBIR	Content Based Information Retrieval
CBVIR	Content Based Visual Information Retrieval
CT	Computer Tomography
DICOM	Digital Imaging and Communication in Medicine
ERSP	Extreme Random Subspace Projection Fern
FLANN	Fast Library for Approximate Nearest Neighbors
HE	Hamming Embedding
HIS	Hospital Information System
HU	Hounsfield Units
IVFADC	InVerted File and Asymmetric Distance Computation
IVFPQ	Inverted File and Product Quantization
k-NN	k-Nearest-Neighbors
LBP	Local Binary Patterns
LSH	Local Sensitive Hashing
LSHE	Locale Sensitive Hashing for Euclidean distance approximation
MAP	Mean Average Precision
MBR	Minimum Bounding Rectangles
MRI	Magnetic Resonance Imaging
NLP	Natural Language Processing
NN	Nearest Neighbor
PACS	Picture Archiving and Communication Systems
PCA	Principal Component Analysis
PET	Probability Estimation Trees
PQ	Product Quantization
QBIC	Query By Image Content
RF	Random Forests
RIS	Radiology Information System
RLSV	Randomized Locality Sensitive Vocabulary
ROC	Receiver Operating Characteristics
RP	Random Projections
SP	Semantic Profile
VA-files	Vector Approximation Files

CHAPTER 1

Introduction

In the medical field, images of different modalities are used for diagnosis and therapy. The number of images produced in radiology departments is rising and can be in the range of several thousands of records a day for just a single hospital [55] resulting in datasets of up to more than 300 million images [44]. Picture Archiving and Communication Systems (PACS) have been created to manage digitally produced medical images and with Digital Imaging and Communication in Medicine (DICOM), a standard for image communication has been set that allows to store patient information along with the image [55]. The integration of PACS into Hospital Information Systems (HIS) or Radiology Information Systems (RIS) allows to associate even more information to the images [44]. Textual components of such a dataset, such as the digital radiology report, can be searched by utilizing tradition text-based search and indexing methods [24, 23]. However, there are several fields that can be thought of which can take advantage of search methods based on the actual visual information of the images rather than associated tags and text. E.g. teaching, research and diagnostics could utilize tools which allow to search for images, similar to the one chosen by the user [55]. This requirement is addressed by so-called Content Based Image Retrieval (CBIR) Systems [23, 55, 75].

CBIR-Systems in the domain of medical images have to consider special properties of the data such as an additional third dimension of the images. Within the scope of this thesis, two critical components of such a specialized system are developed and evaluated. The motivation for this thesis is the idea to take advantage of large data-sets, recorded in the daily routine of hospitals. Not only ways to utilize these datasets by means of machine learning techniques but also a way to take textual data into account are examined. The following introductory chapter is structured as follows. Section 1.1 “Problem Statement” will introduce the basic problems and requirements addressed by this work. In Section 1.2, the “Contribution of the Thesis” is stated in a concise fashion. Finally, Section 1.3 will give an outline of the chapters of this thesis.

1.1 Problem Statement

The aim of the present work is to provide and evaluate indexing and descriptor learning techniques suitable for content based retrieval of 3-dimensional medical imaging data, recorded in the daily routine of hospitals. Thus, a searchable image database should be developed, which allows the search for interesting regions by providing a query image. The design of the components should consider special properties of medical images. Medical images such as Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) are not only three-dimensional but may also be provided with textual components from which semantic information can be extracted. Especially an attached radiology report describes appearances in the image which may be desired by the user and thus provides semantic information. A way to process and utilize this semantic information in the indexing process should be found and implemented. Note, that the development of a user interface of such a system is not in the scope of this work. This thesis focuses solely on the development of an indexing pipeline.

The searchable index should be developed for a clinical decision support system for radiology. Therefore, the performance of the search process should meet speed requirements of interactive applications.

Indexing and searching of a large scale set of medical image data poses domain specific problems. Interesting regions that cover just small parts of the image, the memory consumption of 3-dimensional images and the efficiency of algorithms are issues which have to be considered.

Accuracy and relevance of the search results are important issues when it is the aim to provide decision-making support. Thus, a way to evaluate the indexing methods regarding their suitability to retrieve anomalies in real world images has to be developed and implemented. The major tasks for this work can be summarized in three points:

1. **Feature extraction and descriptor learning:** This component should find a way to extract numerical representations of images in order to make them comparable and searchable. It is not in the scope of this work to compare different feature extraction methods. Is it possible to modify and optimize a single promising method to meet the stated requirements? Furthermore, are there ways to utilize information extracted from radiology reports and to gain advantage of large datasets?

2. **Indexing and Search:** Is it possible to manage and further processes feature vectors in a way to make them searchable in reasonable time? A comprehensive literature search should be conducted and suitable methods should be identified.

3. **Evaluation on real-world data:** A way to evaluate the components of points (1) and (2) has to be found. Evaluation metrics to judge the feature vectors ability to encode clinical relevant visual information have to be identified and applied. All experiments should be conducted on the basis of real world scenarios.

1.2 Contribution of the Thesis

The contribution of this thesis is the development of critical components of a CBIR-System for texture indexing and retrieval of real-world medical imaging data. Existing techniques are

described and discussed with respect to the problem stated. Techniques are adapted and implemented to process real-world test data. The implemented components include feature extraction, descriptor learning, indexing/encoding and an evaluation framework to perform comparable retrieval scenarios. The main contributions are summarized in the following:

- An existing texture descriptor method is adapted to meet speed requirements for large-scale data processing. To make that possible, a critical processing step is replaced by an alternative method.

- Noisy textual information and texture descriptors, both extracted from clinical real-world data, are used to enhance the texture descriptor by a weakly supervised machine learning method. At the end of this learning process, a novel image descriptor, a so-called *Semantic Profile*, incorporates visual and textual information.

- A comprehensive survey on existing indexing and encoding techniques for efficiently storing and searching visual descriptors is given. Several techniques are explained in detail and discussed with respect to the problems addressed by this thesis.

- An evaluation of the methods developed and implemented is performed on real-world data. In order to make that possible exemplary information need is defined and extracted. Possible effects on the retrieval process caused by the techniques used are studied. Qualitative and quantitative evaluation is performed on the *Semantic Profile* descriptor and for different encoding techniques.

1.3 Outline of the Thesis

In the following, the chapters of the thesis are listed and their content is summarized concisely.

- **Chapter 1:** The present chapter provides a general introduction as well as background, problem statement, scope and a preparatory summary of the work.

- **Chapter 2:** Introduces CBIR and describes basic components of a CBIR-System in order to provide the technical context for the rest of the thesis. The focus is on comprehensibility rather than completeness or detailed descriptions.

- **Chapter 3:** Describes the *The Bag of Visual Words Approach for Texture Descriptors*. An entire chapter is devoted to this method, as it serves as the starting point of the indexing pipeline and the input of the image descriptor learning technique developed.

- **Chapter 4:** Describes the *Random Ferns* method and its applications. The chapter is motivated by the fact, that the algorithm of random ferns is an integral part of this thesis and applied to solve several different problems in performance optimization of the Bag of Visual Words approach, weakly supervised learning of image descriptors and the generation of compact codes for indexing.

- **Chapter 5:** Gives an overview on existing indexing approaches. Several methods are described and discussed with respect to this thesis.
- **Chapter 6:** Proposes a novel method for image descriptor learning. A way to combine textual information with visual information by means of weakly supervised learning is proposed. The method and its processing steps are described and discussed in detail.
- **Chapter 7:** Describes the application of promising indexing methods on the descriptors proposed in Chapter 6. As an alternative to existing binary code learning techniques, a way to generate compact codes by means of Random Ferns is proposed.
- **Chapter 8:** Describes the evaluation framework to assess the methods used. Test-data, information need and suitable measures are examined and defined.
- **Chapter 9:** Describes the experiments conducted and shows the results. The ability of the descriptor and indexing methods to rank relevant images is evaluated.
- **Chapter 10:** Concludes the thesis with a summary, ideas for future work and improvements of the methods developed.

CHAPTER 2

Basic Concepts for Content Based Image Retrieval

Content Based Image Retrieval (CBIR) or Content Based Visual Information Retrieval (CBVIR) [55] is the task of finding similar images in a database for a given query image [75]. This chapter will give an introduction to CBIR systems along with an overview of basic components, core techniques and related problems with the focus on the domain of medical images. Due to the fact that multimedia information retrieval and CBIR as a part of it, is a multidisciplinary field of research [79], this chapter can only cover a part of the CBIR associated aspects and makes no claim of completeness. This chapter concentrates in particular on computer vision, information retrieval and software engineering related matters. In the following, a comprehensive introduction is given. Section 2.2 addresses one of the core problems of Information Retrieval, the "semantic gap". Section 2.3 "Architecture and Components" will examine core components of a CBIR-system.

2.1 Introduction

While the first CBIR-systems existed in the early 1980s [55], the first commercial products such as IBM's Query By Image Content (QBIC) came up in the 1990s [55]. The main characteristic of CBIR is, that the search is based on image content properties such as shape, color or texture rather than on textual tags or descriptions [75]. Thus, the queries are sketches or images rather than text [75]. In common CBIR systems, this content-similarity is not measured on the objects directly, but on image descriptors which represent abstractions of the objects [23, 20]. Image descriptors are abstractions of shape, color or texture properties called features [20]. Feature extraction algorithms encode those features to vectors called *feature vectors* [23]. For an image descriptor, a similarity measure is given by a matching function, which utilizes the image feature vectors to give a degree of similarity between two images [23]. The distance is calculated by means of an inverse function where the larger the distance, the less similar the images are.

A reasonable approach of storing collections of images in a database is to represent them as their feature vectors. For a search in the database, the features of the query image have to be extracted and the distances to all vectors in the database have to be computed. The vectors with the smallest distance represent the most similar images. [75]. Thus, whatever the application domain is, CBIR-systems have to provide a means for

- Describing and recording the image content based on pixel/voxel information (image features/descriptors) and
- Assessing the similarity between the query image and the images in the database. [1]

Muller et al. identified three medical domains which can take advantage of CBIR technology [55]:

- *Teaching* - Searching tools for example, can be utilized to find interesting or important cases of medical images in large image repositories which can be presented to students.
- *Research* - Scientists can use mining tools to discover relationships among images and other kind of data like text.
- *Diagnosis* - CBIR searching tools can be utilized in clinical decision support systems. But systems used as a diagnostic aid not only need to prove their performance but they also need to be accepted by the clinicians as a useful tool. Another challenge would be the integration of such a system into daily clinical practice [55].

2.2 The Semantic Gap

"*The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation*" [75].

That is how Smeulders et al. [75] phrase the the term *semantic gap* for visual search. However, the problem of the semantic gap has its roots in text search engines. Castelli and Bergmann [20] give an example of the semantic gap in text-based retrieval. One may consider the task of finding humorous sentences from a digital archive. While this task is simply impossible for a standard textual, syntactic database system, the same system will accept queries such as "find me all the sentences including the word 'steamboat"'. CBIR aims to support the bridging of the semantic gap relying on visual similarity for judging semantic similarity. On one side is a user, looking for images containing certain objects or conveying a certain message, while image descriptors on the other side are based on data-driven features. Those two views may be disconnected [75]. This problem can be tackled by associating higher level semantics to data-driven observations [75]. The most basic way of semantic characterization is to manually attach keywords or captions in which case the problem would be reduced to information retrieval [75]. Though, cost and coverage are two serious objections to that approach. On the cost side, labeling thousands of images is a cumbersome and expensive job. On the coverage side, labeling is

seldomly complete and often context sensitive [75]. Furthermore, requests have to be formulated and their semantics have to be captured by labeling. Relevance Feedback (RF) is another way to alleviate the semantic gap problem [23]. This iterative approach aims to improve the effectiveness of retrieval systems in three steps [23]

- Initial search for a user-supplied query pattern resulting in a small number of result images
- Indication by user which of the retrieved images are relevant
- Automatic reformulation of the query based upon the relevance judgment of the user

These steps can be repeated iteratively until the user is satisfied. In this way, the system learns the user's image perception. Implementations of RF need to be fast enough to support real-time user interaction and needs an integrated learning mechanism [23]. As an alternative to manual labeling or RF, a third way of inferring the information desired to develop more representative features is by exploiting learning approaches or statistical models to represent the semantics [37]. However, in order to make use of such an approach, a corpus of suitable training data has to be available.

2.3 Architecture and Components

Torres et al. [55] identified a basic architecture for CBIR-systems shown in Figure 2.1. Archiving and indexing the images, the extraction of visual features and methods for similarity measurement along with an interface for query specification and result visualization are stated as the core components. This section will describe basic ideas and methods for those components. Due to a lack of relevance for this work the components for visualization and query specification are omitted.

2.3.1 Feature Vector Extraction

The *Feature Vector Extraction* component aims to represent the information contained in image pixels in the form of image features. In other words, this component seeks to bridge the gap between the visual content and its numerical representation [1]. There can be distinguished between two types of visual features:

- *Photometric features* are derived directly from raw pixel intensities and exploit information of color and texture [1].
- *Geometric features* extract shape-based information that can be deducted directly from images but cannot be represented by color or texture [1].

Color and Gray-scale Features

Color photographs are used for diagnosis in ophthalmology, pathology and dermatology [1]. Color is also utilized to scale flow velocities and in nuclear cardiology [1]. A global characterization of an image can be obtained by building a histogram of the pixel color components

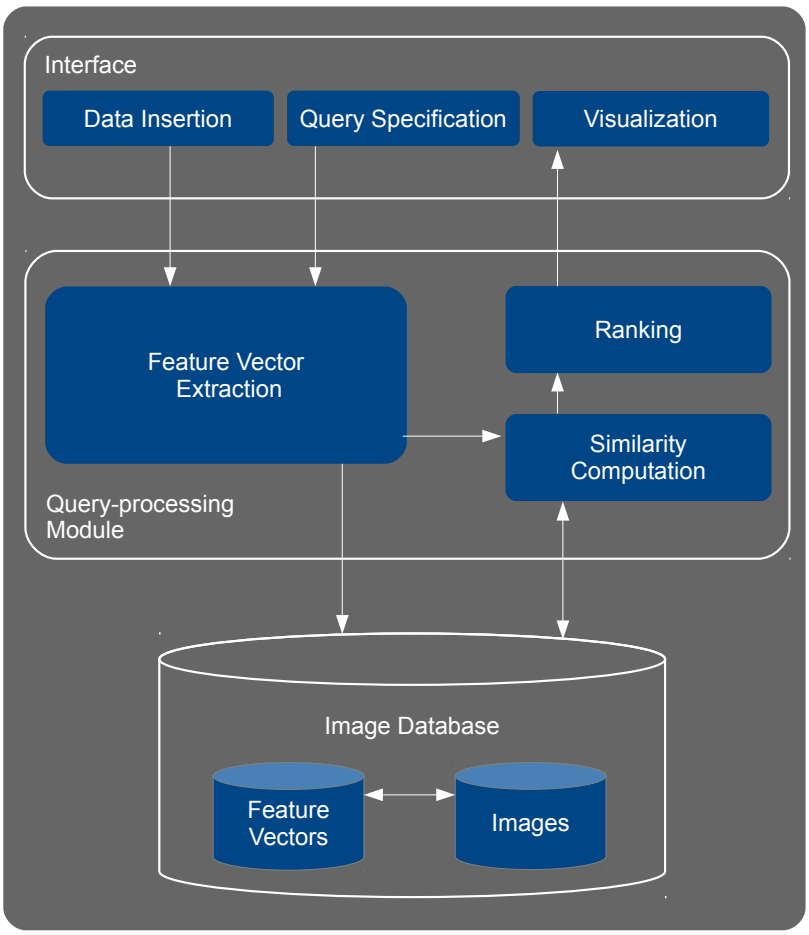


Figure 2.1: Typical architecture of a CBIR systems identified by Torres et al. supporting the two main functionalities *query processing* and *data insertion*. The insertion system is responsible for extracting appropriate features and storing them into the image database. The query processing uses the same feature extraction methods and utilizes some similarity calculation to deliver a ranked result set. (Figure adapted from [23])

or by representing sub-blocks of the image by an average color component vector [1]. Despite the fact that there are several medical imaging methods producing color images, most medical images like CT and MR are gray-scale [1] and color features are not useful for the majority of medical images [1]. A very basic type of feature vector for such a modality are gray-scale histograms [33]. Intensity histograms of CT images are a good example of a feature vector that can represent actual semantic content of an image, perceptible by humans. A CT-scanner puts out integer numbers representing the attenuation coefficient of tissue [42]. The values of those integer numbers are in the range of -1000 to 3000 and have been given the name Hounsfield

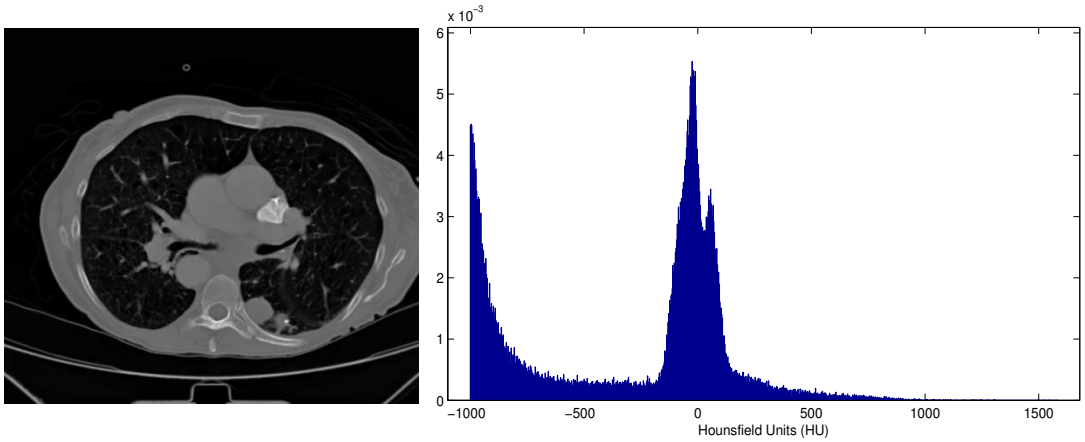


Figure 2.2: A medical CT-image (left) and its normalized intensity-histogram (right). Due to the fact that different kinds of tissue manifest itself in different Hounsfield units [43], a relationship between the histogram and the image content can be drawn for the medical domain.

Tissue	**Range of Hounsfield units**
Air	-1000
Lung	-500 to -200
Fat	-200 to -50
Water	0
Blood	25
Muscle	25 to 40
Bone	200 to 1000

Table 2.1: Certain types of tissue can be associated with a certain range of HU [43]. This table shows a small selection to give some examples. (Data according to [27])

Units (HU) [42].

$$H = \frac{\mu - \mu_{water}}{\mu_{water}} \times 1000 \tag{2.1}$$

Equation (2.1) shows the relationship between the attenuation coefficient and the the corresponding HU where μ_{water} is the attenuation coefficient of water. Thus, a value of $H = 0$ corresponds to water and a value of $H = -1000$ corresponds to air because air has the attenuation coefficient of $\mu = 0$ (if a scanner would be perfectly calibrated). For the domain of medical images, relations between image content and certain ranges of HU can be drawn [43]. Table 2.1 shows some examples of associations of tissue and HU. Figure 2.2 shows an example of an intensity histogram of a CT image based on its Hounsfield scale.

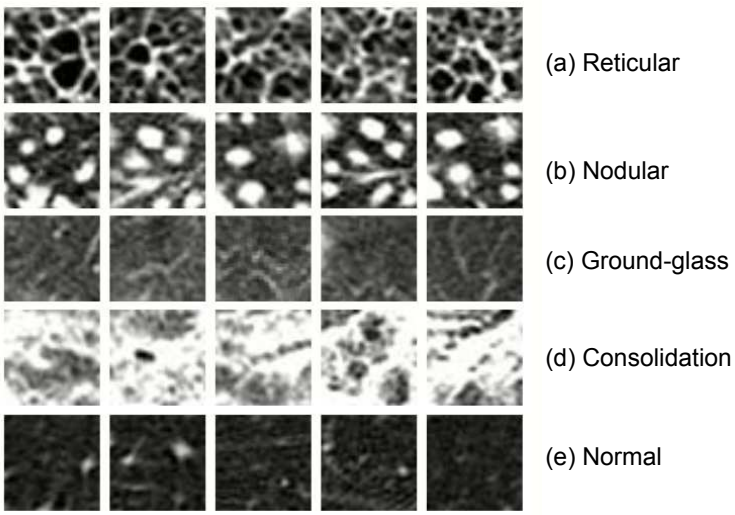


Figure 2.3: Different textures of lung anomalies (a-d) and normal lung tissue (e). (Figures taken from [49] with permission of the author)

Texture Based Features

Texture features aim to encode spatial organization of pixel values [1]. In other words, texture measures capture image characteristics by encoding gradients in different directions of parts of the image [55]. This is useful for images with regions of homogeneous texture [55]. Figure 2.3 shows different textures of lung anomalies compared to a normal lung texture. Wavelets, Gabor filters, co-occurrence matrices or Fourier transformations are some texture encoding techniques mentioned in the context of CBIR [55, 1]. Burner et al. used texture features in a CBIR system to retrieve similar regions in lung CT images [18]. In this thesis, a texture feature extraction method based on the work of Burner et al. is developed. Chapter 3 describes this *Bag of Visual Words Approach for Texture Descriptors* in detail.

Shape Based Features

Rubin et al. use the term *shape* to refer to the "information that can be deduced directly from images and that cannot be represented by color or texture" [1]. So that information called "shape" defines a complementary space to color and texture. Shape can be represented through perceptually grouped geometric cues like edges, contours, joints, polylines or polygonal regions [1]. However, to extract a shape based representation of an image content point sets, contours, curves, regions and surfaces are used as shape-based features [1]. As this kind of representation is not available in the data directly, region-of-interest detection and segmentation are utilized to extract that information from the pixel intensity values [1]. Shape based features have been successfully used for CAD [66]. However, the segmentation problem is stated as the main obstacle towards the use of shape features in CBIR [1].

Local and Global Features

Texture features as well as any other features based on pixel intensity can be extracted from the complete image (global) or on a local level on parts of the image [55, 20]. The basic idea of local feature extraction is to use a partitioning or extraction of patches of the image to cover local properties. Global features can be seen as a special case of local features which are calculated from the entire image. One approach to accumulate locally extracted features is to quantize them into a histogram [75] like it is done in the visual-word paradigm of Sivic et al. [74] following the idea that images can be represented by sets of typical patches of local appearance, called visual words.

If a user starts searching with a whole query image, he or she might be willing to specify a portion of the image to be of interest. This concept has been termed region-based querying [26]. Carson et al. [19] propose a system, which performs image-to-image matching, letting the user select one or more homogeneous regions. In order to make that possible, the images are segmented and each segment is represented by its own feature vector. Burner et al. presented a retrieval method based on local texture features and showed, that their method outperforms global features when searching for certain lung anomalies [18].

Semantic Features and Meta Data

In addition to purely quantitative visual features of the image content, semantic image features or meta data can be inferred from lower abstraction levels [20] or human experts [1]. Radiologists for example describe a variety of information in medical images that describes essential semantics of the image content [1]. Furthermore, medical images can be associated with related information like laboratory reports, clinical diagnosis or demographic information about the individual from which the image was obtained [1]. However, to support content based retrieval on different levels of abstraction, not only quantities that describe the characteristics of interest must be defined but also methods to extract those. Appropriate similarity measures have to be found and indexing techniques to efficiently search large collections of data have to be adopted [20].

2.3.2 Similarity Calculation

The (dis)similarity of a pair of images is computed by comparing the feature vectors extracted from the images [75]. A reasonable approach is to interpret the feature vectors with d coefficients as points in a d-dimensional space, the *feature space* [55]. The dissimilarity can be measured by computing a distance between the vectors in the feature space [75]. Note, that this section does not provide a summary of all possible metrics and does not discuss their suitability for different image descriptors or indexing methods but gives an introduction to basic similarity measures.

Datta et al. summarized some key motivating factors behind the design of a similarity measure [26]:

- Agreement with semantics
- Robustness to noise

L_p **Minkowsky family**	
(1). Euclidean L_2	$d_{Euc}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{d} \lvert\mathbf{x}_i - \mathbf{y}_i\rvert^2}$
(2). City Block L_1	$d_{CB}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{d} \lvert\mathbf{x}_i - \mathbf{y}_i\rvert$
(3). Minkowski L_p	$d_{Mk}(\mathbf{x},\mathbf{y}) = \sqrt[p]{\sum_{i=1}^{d} \lvert\mathbf{x}_i - \mathbf{y}_i\rvert^p}$
(4). Chebyshev L_∞	$d_{Cheb}(\mathbf{x},\mathbf{y}) = \max_i \lvert\mathbf{x}_i - \mathbf{y}_i\rvert$

Table 2.2: Definitions of different L_p metrics for the distance calculation of two points $\mathbf{x}$ and $\mathbf{y}$ in a d-dimensional space. While the Euclidean distance (1) can be seen as the beeline between two points, the city block distance (2) would be the way a taxi would take in a city with orthogonal streets. Equations (1) and (2) are generalized to the Minkowski distance metrics (3). (Table adapted from [21])

- Computational efficiency (e.g. the ability to work in real time and on large scale)
- Invariance to background (e.g. allowing region-based querying)
- Local linearity (i.e. following triangle inequality in a neighborhood)

If two images are represented by two vectors $\mathbf{x}$ and $\mathbf{y}$, the distance between these vectors is used as dissimilarity measure [26]. This distance can be calculated utilizing different metrics. Among others, the so called L_p metrics or Minkowsky metrics are stated in the literature as typically used in image retrieval [26, 20, 15, 55, 1, 75, 23]. Especially the Euclidean distance (L_2) is supported by indexing methods used in image retrieval [15, 41, 4]. Table 2.2 shows the definitions of different L_p metrics. To illustrate the differences, Figure 2.4 shows two-dimensional unit-balls according to different values of p. Note that $p = 1$ is the *City Block*, $p = 2$ the *Euclidean* distance. If additional weights need to be assigned to the dimensions of a feature vector, the weighted Euclidean can be defined as follows

$$d_{wEuc}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{d} w_i.\lvert\mathbf{x}_i - \mathbf{y}_i\rvert^2} \tag{2.2}$$

where $w_1, ..., w_d$ are the weighting factors for the dimensions [15]. In the 2-dimensional case, the unit-ball would result in an ellipse for two differently weighted dimensions.

The chosen metric effects both, effectiveness and computational complexity of the retrieval. The effectiveness indicates to which extent the quantitative similarity match the perceptual, subjective one. Some histograms have the property, that the difference between large bins is less

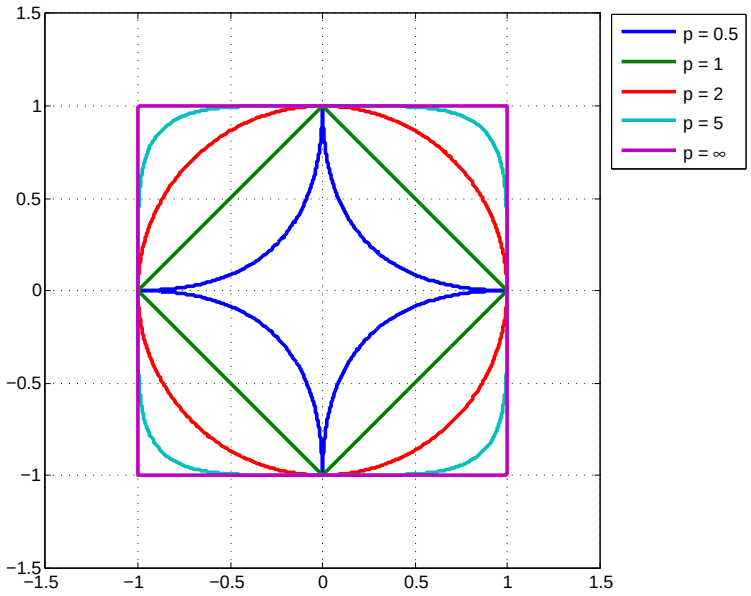


Figure 2.4: Different unit balls (distance from the center to every point of the colored lines is 1) in a 2-dimensional space graphed for different values of p. For $p = 2$ (Euclidean distance) the unit ball is just a circle. $p = 1$ is the city block distance. $p = \infty$ is the Chebyshev distance. $p = 0.5$ and $p = 5$ are two examples of other Minkoswky metrics.

important than the difference between small bins [63]. A metric which takes this into account is the Chi-Squared (χ^2) distance which is defined as follows [63]:

$$\chi^2(\mathbf{x}, \mathbf{y}) = \frac{1}{1} \sum_{i}^{d} \frac{(\mathbf{x}_i - \mathbf{y}_i)^2}{\mathbf{x}_i + \mathbf{y}_i} \tag{2.3}$$

According to Pele [63], the χ^2 metric was successfully used for texture and object categories classification, near duplicate image identification, shape classification and others.

2.3.3 Storage and Indexing

In the sections above, it has been shown that the content of an image can be represented by a feature vector or a set of vectors. Application of these extraction methods on all images in a dataset yield a set of feature vectors which we can call the *data file*. We have examined ways of how two of these feature vectors can be compared to derive a similarity measure. To search the data file for a query image, a feature vector has to be computed and compared with the elements in the data file on the basis of the similarity function. With increasing size of the dataset, computational performance can not be ignored as an issue. Especially for interactive systems, response times of less than one second are often specified as a usability requirement [76].

If we assume that the feature vectors are stored in a standard linear file with one record to each of N images, the search is bound to scan through all vectors [75]. In that case, N fetches of a record are required plus subsequent calculation of the distance to find the most similar vector to the query vector. The run time of this search method is in $O(N)$ [75]. This approach may not be feasible if a database has to store up to billions of images [2] but has to meet certain runtime requirements. Not only the number of images has to be considered but also the cardinality of the feature vectors. While for low (1-4) and moderate (4-8) dimensional spaces, well-working methods are known, their performance is degrading as the number of dimensions is increasing [80]. This phenomenon has been termed as the "dimensional curse" [80]. The curse of dimensionality is discussed in detail in Section 5.2.3. The dimensionality of feature vectors depend on the method which produced the vector and can vary from moderate to large with several hundred up to more than 1000 dimensions [80].

To achieve faster retrieval times compared to exhaustive search, methods solving the Approximate Nearest Neighbor (ANN) search problem can be used. The key idea of ANN is to find the nearest neighbors with a probability <1, instead of finding the exact nearest neighbors with probability 1 by sacrificing a certain amount of accuracy for higher performance in retrieval speed and memory consumption compared to exact methods [41].

One idea to deal with high dimensions is to reduce the dimensionality of the vectors by methods such as the Principal Component Analysis (PCA) [73, 69] or Random Projections (RP) [13]. Once the vectors are in a lower dimensional space, tree based methods, such as the kd-tree can be used [65]. Nevertheless, the reduction of dimensionality can be accompanied with loss of information [73] and therefore can influence the quality of the retrieval results.

Chapter 5 examines several state of the art methods for exact and approximate indexing of medium to high dimensional feature vectors.

2.4 Discussion

In this chapter, an introduction to CBIR is given. The decision for a certain architecture of such a system and the methods used are based on the systems requirements. A decision for a certain feature of a CBIR-application influences the architecture of the system. E.g. the implementation of a relevance-feedback feature, not only requires fundamental change of the user interface but also indexing and learning have to be adapted and the usability of the system is influenced. To use partitions of an image instead of a global representation allows to query for regions but increases the size of the index considerably because a descriptor for every single partition has to be stored and managed. The decision for a certain feature extraction method effects the cardinality of the feature vector and thus again the requirements for indexing. Image descriptors relying on visual information only have no association to clinical relevance. This thesis tackles the problem of reducing this semantic gap for a texture based descriptor by using information extracted from radiological reports. Suitable indexing methods are identified for the descriptor developed and incorporated in into an indexing framework.

CHAPTER 3

State of The Art: The Bag of Visual Words Approach for Texture Descriptors

This chapter examines a method to generate texture descriptors based on the Bag of Visual Words approach. As mentioned in Section 2.3.1, texture descriptors are used to encode spacial organization of pixels respectively voxels in an image. In this work, texture descriptors are used as the basic image features to facilitate content based retrieval. The Bag of Visual Words Approach (BoVW) (Section 3.1) is a paradigm with its roots in text retrieval systems and serves as the basic framework for the examined texture descriptor. Section 3.2 examines Local Binary Patters (LBP), a way to describe local texture information. Section 3.4 describes the vocabulary- and histogram building process, two elementary steps towards BoVW. An oversegmentation and multi-scale approach of the examined technique to cover more texture properties is described in Section 3.3. Finally, the potential of the examined method to be used in this thesis is discussed in Section 3.6.

3.1 The Bag of Visual Words Approach

Bag of Visual Words are introduced by Sivic et al. [74] and are based on text retrieval techniques adapted for visual content. Basically, this approach counts the occurrence of words in a text in order to make it comparable to other texts. In the following, processing steps to index text files are listed according to [74].

- The documents are parsed to obtain a representative vocabulary
- Words are represented by their stems
- Stop lists are introduced to reject very common words

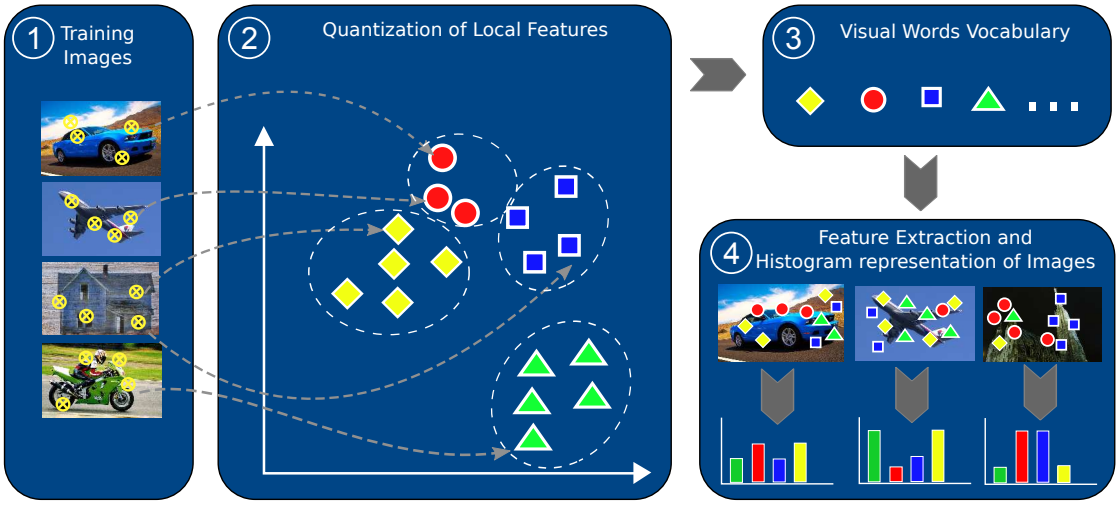


Figure 3.1: The Bag of Visual Words approach. (1) Local features are extracted from a set of training images. (2) Training features are quantized to obtain uniform representations. (3) The quantized features are called "visual words" and the whole set of words form the "visual vocabulary". (4) The learned vocabulary is used to describe an image as vector given by the frequency of visual words.

- Words are assigned to a unique identifier
- Documents are represented by vectors of the frequencies of occurrence of the words.

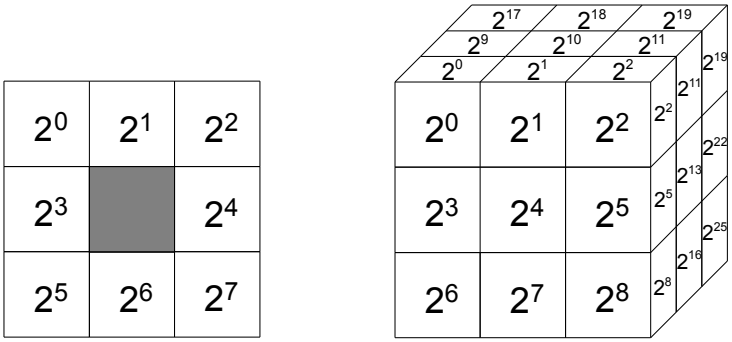
The vectors obtained are used for retrieving similar text documents by applying some distance metric. However, in the visual domain, a visual vocabulary has to be learned by extracting and quantizing a training set of local features. In other words, local feature-sets of varying candidates are quantized to obtain uniform representations, referred to as a "visual vocabulary" and then summarized into histograms. By doing so, the inherent structure in the visual data is learned [18]. Figure 3.1 shows the basic steps of the BoVW approach. The vocabulary is trained on a set of images called the "Training Set". Local features are extracted and quantized to form the visual vocabulary.

3.2 Voxel based Texture-Descriptor by Local Binary Patterns

Lets define $\mathcal{V} = \{v_1, \ldots, v_V\}$ as the set of all voxels of a volume. A voxel-based desciptor is a mapping function

$$D : \mathcal{V} \to \mathbb{R}^d, \tag{3.1}$$

so that $v_i \mapsto \mathbf{d}$. In order to use the BoVW approach for textures, a local descriptor capturing local gray value changes can be used [18]. This section describes such a descriptor used in the medical domain. The original two-dimensional version of Locale Binary Patterns (LBP) is described by Ojala et al. [58]. Given a pixel i, the LBP operator is computed as follows:



(a) (b)

Figure 3.2: (a) The two-dimensional LBP descriptor operates on 3x3 pixel patches. An 8 bit number encodes the gray-value differences between the center pixel and its neighbor pixels. (b) The thee-dimensional LBP3D descriptor by Burner et al. encodes the gray value changes of a 3x3x3 voxel cube to a 26 bit number. (Figures taken from [18] with permission of the author)

1. Value independent weights are assigned to each neighbor pixel
2. The value of i is compared to each neighbor pixel
3. All weights of neighbor pixels with a higher value than the value of i are summed up

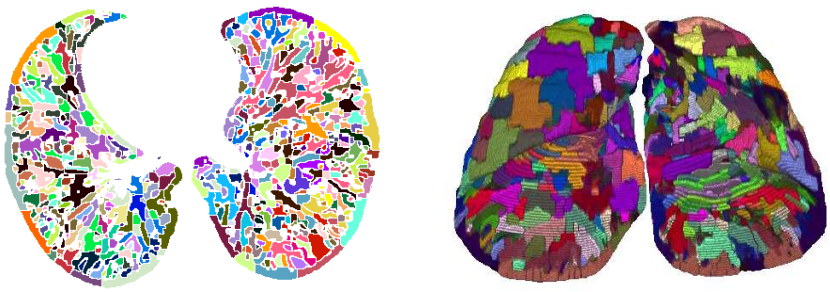
Figure 3.2a shows the weights used for a center pixel. The computation of the LBP operator yields an 8-bit $(3^2 - 1)$ value from 0 to 255 that describes the gray-scale structure of adjacent pixels in a 3x3 pixel grid [18]. Burner et al. developed a three-dimensional LBP texture descriptor (LBP3d) to capture lung tissue characteristics in CT images [18]. In difference to 8-bit LBP values, extracted from 3x3 pixel patches, LBP3d generates 26 bit $(3^3 - 1)$ descriptors on 3x3x3 voxel cubes (Figure 3.2b). The additional dimension increases the number of possible texture units from 256 to $\approx 67.1 * 10^6$. To incorporate local contrast and gray value intensity, Burner et al. [18] incorporate a local contrast measure and the average intensity of the voxel cube resulting in the LBP3d/CI descriptor defined as follows:

$$\mathbf{d}_{LBP3d/CI} = [\mathbf{d}_{LBP3d}, \beta\mathbf{d}_C, \gamma\mathbf{d}_I], \tag{3.2}$$

where $\mathbf{d}_{LBP3d}$ is the three-dimensional LBP operator, $\mathbf{d}_C$ is the contrast measure and $\mathbf{d}_I$ is the average intensity. The factors β and γ allow to configure the impact of the two additional measures. In total, the descriptor has 28 dimensions, 26 LBP bits, one contrast dimension and one intensity dimension.

3.3 Oversegmentation and Multi-Scale Approach

Considering CBIR of medical images like lung CT, one may be interested in local regions rather than the entire image. One way to overcome this problem is to partition the image into small



(a) (b)

Figure 3.3: Oversegmentation of a lung volume by the algorithm of Wildenauer et al. [82]. (a) A 2D cut and (b) a 3D rendering of supervoxel partitions, extracted from a CT image. (Figures taken from [18] with permission of the author)

patches [82]. Wildenauer et al. propose an oversegmentation algorithm to partition an image according to its texture areas, resulting in unshapely voxel patches with high (compared to random partitioning) texture homogeneity called "supervoxels" [82]. Burner et al. [18] apply this algorithm on the medical domain. In context of the BoVW approach, each of the supervoxels can be treated as a volume by its own and a word histogram has to be built for each supervoxel. Figure 3.3a shows an axial cut of an oversegmented lung and Figure 3.3b shows a three-dimensional rendering of an oversegmented lung.

In addition to oversegmentation, Burner et al. [18] implement a multi-scale approach for their descriptor in order to capture texture structures in different resolutions. The motivation behind this approach is based on the work of André et al. [3] who developed a BoVW descriptor to classify microscopic images. André et al. extracted their descriptor on different scales to capture microscopic as well as mesoscopic (group of cells) features in microscopy images [3]. Burner et al. extract the LBP operator on different scales to capture the granularity of texture and to reduce scale variance. The images are down-sampled by an image pyramid so that each scale step reduces the volume size by one quarter. The feature extraction and vocabulary learning is performed independently for each scale. As a last step, the final image descriptor is constructed by concatenating the texture word histograms. Figure 3.4 gives an overview of the described multi-scale approach.

3.4 Vocabulary Building and Histogram Representation

Let $\mathbf{d} \in \mathbb{R}^d$ denote a local descriptor extracted from a training image. A vocabulary $\mathcal{W}$ can be seen as a set of W integers where each value represents a group of descriptors $\mathbf{d}$ which are similar (e.g. build a cluster) in the descriptor space. Thus, we are looking for a function

$$V : \mathbb{R}^d \rightarrow \{1, \ldots, W\} \tag{3.3}$$

which allows a mapping $\mathbf{d} \mapsto w$ for $w \in \mathcal{W} = \{1, \ldots, W\}$.

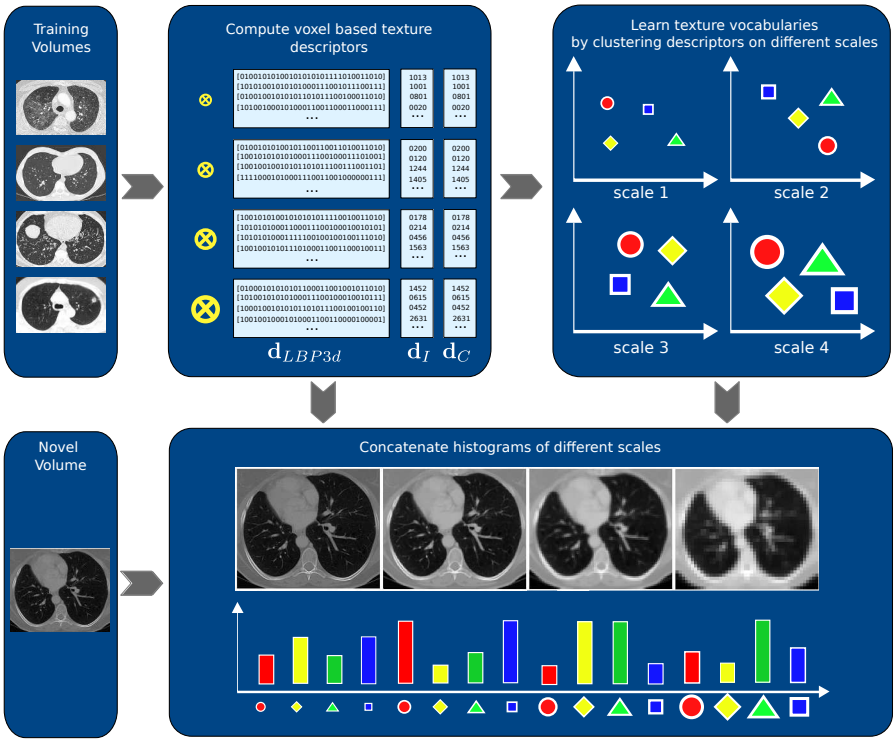


Figure 3.4: A multi-scale approach for BoVW [18, 3]. The voxel based texture descriptors LBP3d/CI are extracted on different scales. Vocabulary learning is performed independently for each scale. For a novel image a word histogram is generated on each scale. The final descriptor is constructed by concatenation of the histograms. (Parts of the figure according to [18])

One way to generate a vocabulary is to interpret the descriptors as vectors in an Euclidean space and apply k-means clustering with W cluster centers, resulting in a partitioning of the feature-space [18].

3.4.1 k-means Clustering

K-means is a clustering algorithm which has to be initialized with a fixed number of cluster centers. As an iterative procedure, k-means minimizes the sum of squared errors within the clusters [46]. K-means clustering in the context of vocabulary building can be seen as the optimization of an objective function given by:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{W} r_{nk} \parallel \mathbf{d}_n - \mathbf{w}_k, \parallel \tag{3.4}$$

where N is the number of training descriptors and W is the target size for the vocabulary. The binary indicator variables $r_{nw} \in \{0, 1\}$ describe to which word w, descriptor $\mathbf{d}_n$ is assigned to. This function represents the sum of squared errors (sum of squared distances of each descriptor

to its assigned word). After random initialization of $\mathbf{w}_k$, the objective function is optimized by an iterative procedure corresponding to successive optimization with respect to r_{nk} and $\mathbf{w}_k$ [14]:

1. J is minimized by changing the cluster assignments r_{nk} with fixed $\mathbf{w}_k$
2. J is minimized by changing the cluster centers $\mathbf{w}_k$ with fixed r_{nk}

These two optimization steps are repeated until convergence or until a pre-defined number of iterations have been performed. The result is a Voronoi partitioning of the feature space. Each partition represents all possible variations of one word. To assign a novel descriptor to a word, a nearest neighbor search among the cluster centers $\mathbf{w}_k$ is performed. Finally an image $\mathbf{I}$ or a supervoxel can be described by the histogram $h(\mathbf{I})$ of words w it contains resulting in a visual-word feature vector.

3.4.2 Visual-Vocabulary Size

Unlike in text indexing, a visual-words vocabulary has no natural fixed size. It is solely controlled by the number of clusters chosen in the vocabulary learning process. Defining the right vocabulary size involves the trade-off between its discriminating power and generality [83]. In the following, the terms "small" and "large" are used as relative expressions and should be seen in connection to application and data specific properties. By using a small vocabulary, the visual-word feature poses the risk of being not discriminative enough, as dissimilar and thus discriminating local-descriptors can map to the same visual word [83]. With increasing vocabulary size, the visual-word descriptor becomes more discriminative but less general and in that sense more sensitive to noise since similar local-descriptors can map to different visual-words. The vocabulary size also influences the cost of vocabulary learning and computation of the visual-word features. In literature, visual-vocabulary sizes from hundreds to tens of thousands are reported [83].

3.5 Distance Metric for Visual-Word Features

Different, general and universal distance metrics have been examined in Section 2.3.2. However to calculate the similarity between word histograms, more sophisticated similarity measures are reported in literature. Sivic et al. applied a weighting scheme to the visual-words according to text retrieval techniques [74]. A standard weighting in text retrieval is the "term frequency-inverse document frequency" (*tf-idf*) which is defined as follows. Given a word frequency vector (word histogram) of length W for a document, according to [74] the weighting components $(\vartheta_1, ..., \vartheta_w, ..., \vartheta_W)$ are given by

$$\vartheta_w = \frac{n_{wd}}{n_d} \log \frac{N}{n_w} \tag{3.5}$$

where n_{wd} is the number of occurrences of word w in document d, n_d is the total number of word in document d, n_i is the total number of occurrences of word i in the whole database and N is the number of documents in the database [74]. Less formal, Equation 3.5 increases the words weighting for terms occurring often in a particular document as it is supposed to describe

it well but decreases its weighting factor when it appears often in the database. After weighting, Sivic et al. used the normalized scalar product to compare two vectors [74].

Burner et al. used a distance measure, that takes the similarity between texture words into account. The "diffusion distance" algorithm described by Ling et al. [45] uses a weight matrix to consider the relationship between words. Burner et al. chose this method as it increases the robustness of the BoVW method regarding the choice of the vocabulary size W.

3.6 Discussion

This chapter examined an adaption of the BoVW approach with the aim to describe texture in medical images. An implementation of this approach was integrated into a CBIR prototype used to retrieve similar lung regions [18]. The facts that the vocabulary can be trained unsupervised and that the descriptor covers three-dimensional information of the texture structures are the reasons that this descriptor is considered for this work. Burner et al. performed pilot experiments based on the lung anomaly retrieval scenario with a limited dataset where they showed the principal suitability of the approach to distinguish between lung anomaly and unsuspicious texture [18]. As a starting point for our work, we performed initial experiments on real-world data regarding processing time. We use a parallelized C++ implementation of the LBP3d extractor method, developed by Burner et al. [18]. A 12-Core 24-Hyperthread CPU is used to conduct the following experiments. Given a volume with 587x587x547 isotropic resolution, extraction of the LBP features on four scales needs 23 seconds. However to perform the BoVW approach, nearest neighbor search for densily sampled LBP descriptors has to be performed to determine the associated visual words. Even if just every second voxel is used (Burner et al. used every voxel for their experiments), the nearest neighbor search needs 110 seconds for a vocabulary-size of 300. Furthermore, for every scale, an independent vocabulary is built, therefore, if 4 scales are implemented, 440 seconds are needed to determine the texture words for one volume. The nearest neighbor implementation of the *Yael Toolbox*[1] is used to perform the calculations. Therefore, when neglecting processing overhead, 95% of processing time is spent to determine the visual words for the voxel descriptor and just 5% is actually used to extract the voxel based descriptors. Typical BoVW approaches (e.g. scene classification in two dimensions) extract just several hundreds or thousands of descriptors per image [83]. Thus, in order to improve processing time of this descriptor extraction technique, improvement in the descriptor to word assignment offers high (95%) potential.

The examined approach is an example of an exclusively visual data driven feature vector. No relation between the generated numerical representation and radiological categories is given. Oversegmentation leads to an increased number of vectors that needs to be compared for search. Thus, an indexing method has to be used that allows search in reasonable time (e.g. one second for interactive systems).

Besides search time, memory consumption of the index is an issue that can be considered. Assuming a vocabulary size of 300 visual words on 4 scales results in a 1200 single value feature vector. If we consider an index for 10000 volumes with supervoxel size of $1cm^3$ and about 7000

[1] https://gforge.inria.fr/projects/yael/

supervoxels per volume (median number of $1cm^3$ size supervoxels within a lung only, see Table 8.1), an index would need memory of about

$$\frac{(10000 * 7000 * 1200 * 4\ byte)}{1024^3} \approx 313\ gigabyte. \tag{3.6}$$

While this can be stored on secondary memory, the size of the feature vector per supervoxel reduces the number of feature vectors which can be stored in primary memory for search. Furthermore, limited read speed decreases the number of vectors that can be loaded to main memory in a certain time. Thus, depending on the indexing technique used, the reduction of the memory footprint for a supervoxel is an issue that needs to be tackled.

CHAPTER 4

State of The Art: Random Ferns for Classification and Vocabulary Building

This chapter examines the method of *Random Ferns* and how it is applied to solve different problems in the tasks of classification and retrieval. Section 4.1 examines the work of Özuysal et al. who introduced random ferns as an alternative to Random Forests (RF) in order to develop a fast keypoint recognition technique [60, 61]. Section 4.2 examines an adaptation of the random ferns approach as an alternative to k-means clustering to be used in the BoVW approach [62, 52].

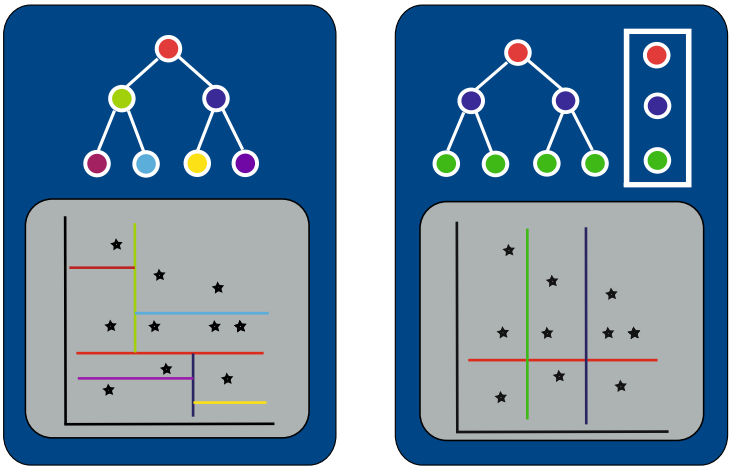
The chapter is motivated by the fact, that the algorithm of random ferns is an integral part of this thesis and applied to solve different problems: In Section 4.3, ideas to apply random ferns on our problems to (1) find a good visual descriptor for medical image retrieval and (2) to find a suitable indexing method are discussed.

4.1 Random Ferns for Classification

Random ferns were originally introduced by Özuysal et al. for fast keypoint recognition [61]. Later-on the method was discussed in more detail in [60]. Fast keypoint recognition is the task of finding texture patches surrounding keypoints across images acquired under widely varying poses and lightning conditions [60]. Such a corresponding is used to register different views of the same scene. The tracking of objects across video frames is one application where keypoint recognition is used.

4.1.1 Keypoint Recognition by Classification

Özuysal et al. interpreded the keypoint recognition task as a classification problem. The changing perspective and lighting conditions of possible patches around an image keypoint are seen as instances of a class so that one keypoint is a representative of a class and the number of keypoints is the number of classes. Random ferns are used as a classifier to find the corresponding



(a) (b)

Figure 4.1: Difference between tree and fern. While a tree (a) is a hierarchical structure, a fern (b) is a sequence of independent decision functions. (Figure according to [62])

keypoint for a novel image patch. The ferns classifier is trained on a database of patches obtained by warping the patches found in a training image by randomly chosen homographies [60].

4.1.2 Patch Recognition by Random Ferns

Random ferns are a simplification of the Random Forests (RF) approach developed by Leo Breiman and Adele Cutler [17]. RF is an ensemble classifier with the basic idea of growing multiple binary decision trees which recursively partition the input entities. A single tree embodies a probability distribution of the input classes. However, as a single tree may not be discriminative enough, a number of trees is used where each of them partitions the input differently. The statistical model of the outcome of the trees is used to determine the class. Özuysal et al. argue, that when the node tests are chosen randomly, the power of the tree based approach derives from combining groups of binary tests rather then the tree structure itself [60, 61]. On the basis of this argument, they replaced the hierarchical trees by nonhierarchical ferns. If $\mathcal{L}$ denotes a binary decision function, the major difference between a fern and a tree is given by its dependence of the outcome of the previous function. In contrast to a tree, a fern $\mathcal{F}$ can be seen as a sequence of binary decision functions:

$$\mathcal{F} = \langle \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_L \rangle. \tag{4.1}$$

Figure 4.1 illustrates the differences of a tree and a fern in two dimensions.

In the following, a formal motivation to use random ferns for classifying a novel image patch is given based on [60]. Let $\mathcal{L}_i,\ i = 1, \ldots, N$, be the sequence of all binary features (binary decisions) that will be calculated on the patch to be classified and let $t_j, j = 1, \ldots, T$ be the set of classes. The aim is to find

$$\hat{t}_j = \underset{t_j}{\operatorname{argmax}}\, P(C = t_j \mid \mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_N), \tag{4.2}$$

with C as random variable. Application of Bayes formula yields

$$P(C = t_j \mid \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N) = \frac{P(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N \mid C = t_j)P(C = t_j)}{P(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N)}. \tag{4.3}$$

Assuming a uniform prior $P(C)$ the problem reduces to

$$\hat{t}_j = \underset{t_j}{\operatorname{argmax}}\, P(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N \mid C = t_j), \tag{4.4}$$

The binary tests used by Özuysal et al. are simple comparisons of pixel intensities in an image patch $\mathbf{I}$ so that

$$\mathcal{L}_i = \begin{cases} 0 & \text{if } \mathbf{I}(d_{i,1}) < \mathbf{I}(d_{i,2}) \\ 1 & \text{if } \mathbf{I}(d_{i,1}) \geq \mathbf{I}(d_{i,2}) \end{cases} \tag{4.5}$$

where $d_{i,1}$ and $d_{i,2}$ are two randomly selected pixels. Due to the relative simple features (comparison of intensities), it is argued that many $(N \approx 300)$ features are needed for good classification and thus the complete representation of the joint probability is infeasible. On the otherhand, assuming complete independence would completely ignore correlation between features. Thus, the compromise, to partition the features into M groups of size $L = N/M$ is suggested and these groups were defined as *ferns*. This seminaive Bayesian approach by modeling only some of the dependencies of the features yields the conditional probability

$$P(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N \mid C = t_j) = \prod_{k=1}^{M} P(\mathcal{F}_k \mid C = t_i) \tag{4.6}$$

with $\mathcal{F}_k \subset \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N\}$ representing the set of binary features extracted by the kth fern. By interpreting these binary features as base 2 number, a ferns response forms an integer so that $\mathcal{F} \in \{1, \ldots, 2^L\}$.

4.1.3 Training Phase

In a training phase, the class conditional probabilities $P(\mathcal{F} \mid C = t_i)$ are estimated for each fern. For a single fern, the probability

$$p_{y,t_j} = P(\mathcal{F} = y | C = t_j), \tag{4.7}$$

for a leaf $y \in 1, \ldots, 2^L$ to represent class t_j is estimated by

$$p_{y,t_j} = \frac{N_{y,t_j} + N_r}{N_{t_j} + 2^L \cdot N_r}, \tag{4.8}$$

where N_{y,t_j} denotes the number of training samples belonging to class t_j evaluate to the leaf value y, N_t is the total number of training samples for class t_j and N_r is regularization term for leafs where no or very few samples evaluate to in the training phase. This regularization is necessary as an empty leaf might just be an artifact caused by the limited size of the training set and a $p_{y,t_j} = 0$ would be too selective.

4.2 Random Ferns for Subspace Partitioning and Vocabulary Building

The Random Ferns method proposed by Özuysal et al. [60] acts as a feature extraction and classification method. However, iterative application of binary tests to partition a feature space in order to construct an implicit dictionary to be used in a BoVW approach is proposed by Pauly et al. [62] and Mu et al. [52]. A definition of a visual vocabulary was given in Equation 3.3. A k-means based approach to learn a visual vocabulary is described in Section 3.4.

As an alternative, Mu et al. [52] propose an approach inspired by RF [71, 51] and Local Sensitive Hashing (LSH) [25] called Randomized Locality Sensitive Vocabulary (RLSV) which shows resemblance to the Random Ferns [60] based approach of Pauly et al. [62] called Extreme Random Subspace Projection Fern (ERSP). It has been shown that both methods can generate similar or even better performing vocabularies compared to k-means when used for classification [62] and retrieval [52] by considerably reducing computation time by factors of more than 8000 [62, 52].
In a vocabulary building process, based on explicit clustering like k-means, a cluster center constitutes a Voronoi cell and each sample feature vector is assigned to its nearest center. A randomized vocabulary is generated by partitioning the feature space by means of applying a sequence of binary decisions to a feature vector. A target visual word for a candidate can be determined by applying those binary tests and no distance computation is necessary. This sequence of binary tests can either be independent [52, 62] by using random ferns, or dependent [71, 51] by using RF. For this thesis, we examine the application of random ferns to generate an implicit vocabulary.

Mu et al. defined the term *Random Visual Word* to describe a vocabulary, which is generated by binary decision functions:

Definition 4.1 ***Random Visual Word:*** *There is a bi-mapping between any visual word w_i and valid permutation π_i from $\{0,1\}^L$. Any two samples $p, q \in \mathbb{R}^d$ belong to the same visual word if for any $i \in \{1, ..., L\}$, there is $b_i(p) \oplus b_j(q) = 0$, where $\oplus$ denotes the XOR bit operation. [52]*

To make that possible a binary decision function operating on a feature vector has to be found so that

$$\mathcal{L}(\mathbf{d}) : \mathbf{d} \mapsto \{0,1\} \tag{4.9}$$

and

$$\mathcal{F}(\mathbf{d}) : \mathbf{d} \mapsto \{0,1\}^L. \tag{4.10}$$

Hence, the response of a fern to a feature vector results in a binary vector. This binary vector encodes the very partition of the feature space the vector is lying in. Note, that the binary codes represent values in the range $(1, \ldots, 2^L)$ so that $\mathcal{F}$ satisfies the requirement for function V in Equation 3.3.

The binary decision function $\mathcal{L}$ can be designed in different ways. Mu et al. used a function in the form of

$$\mathcal{L}_l(\mathbf{d}) = \begin{cases} 0 & \text{if } (\mathbf{s}_l)^\top \cdot \mathbf{d} \leq 0 \\ 1 & \text{if } (\mathbf{s}_l)^\top \cdot \mathbf{d} > 0 \end{cases} \tag{4.11}$$

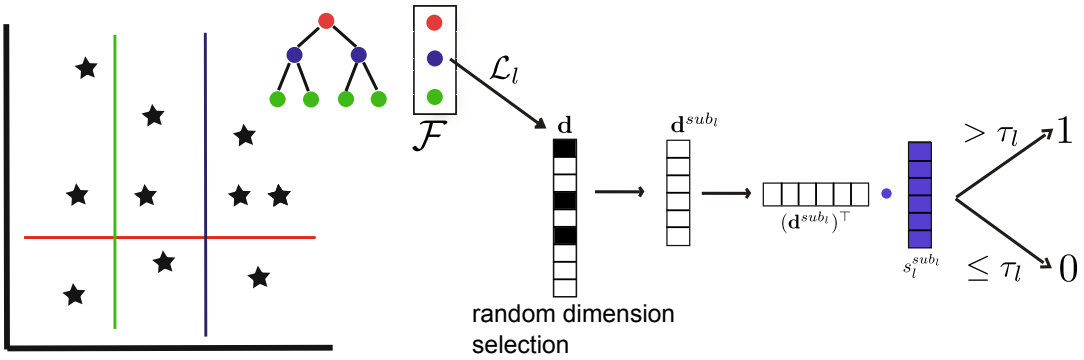


Figure 4.2: Illustration of the splitting function of ERSP. At each node $\mathcal{L}$, subdimensions are randomly selected. Subsequent, the vector is projected onto a random vector before a random threshold operation is applied. (Figure according to [62])

where $\mathbf{s}_l$ is a vector sampled from the unit-hypershpere. This function belongs to the family of LSH and approximates an angle oriented distance [5].

Pauly et al. defined a slightly different splitting function. ERSP uses random dimension selection and random projections at each node. Thus, for a split test, a set of dimensions from the feature space is randomly selected so that the sub vector $\mathbf{d}^{sub_l}$ is considered only. Like in Equation 4.11, the sub vectors are then projected into a one-dimensional space by building the dot product with unit vector $\mathbf{s}_l^{sub_l}$ randomly sampled from the unit sphere so that $(\mathbf{s}_l^{sub_l})^\top \cdot \mathbf{d}^{sub_l}$ is a scalar. To get a binary value, this scalar is compared to a threshold τ_l which can be defined according to the data by using the median of the projected vectors or completely at random [62]. The proposed decision function is defined as follows:

$$\mathcal{L}_l(\mathbf{d}) = \begin{cases} 0 & \text{if } (\mathbf{s}_l^{sub_l})^\top \cdot \mathbf{d}^{sub_l} \leq \tau_l \\ 1 & \text{if } (\mathbf{s}_l^{sub_l})^\top \cdot \mathbf{d}^{sub_l} > \tau_l \end{cases} \tag{4.12}$$

Figure 4.2 illustrates this splitting function.

4.3 Discussion

In this chapter, the method of random ferns has been described and how it is used to solve different problems. The application of random node tests in a sequential rather than a hierarchical way is the core idea of random ferns. The partitioning, induced by ferns has been used for classification and as vocabulary.

Due to the simple structure of random ferns, lower runtimes compared to alternative methods are reported for vocabulary construction [52, 62]. Pauly et al. report a runtime of 1.08 s to induce 2048 random partitions by using 8 ferns of depth 8. The k-means method used, needs 2.3 h to generate 1000 clusters [62]. However, eventhough the partitioning was induced randomly by random ferns the vocabulary performed comparable to a k-means vocabulary used in modality

detection [62]. Mu et al. report similar results using random vocabularies for performing a CBIR task [52]. The reported results make the random ferns method a good candidate to improve the runtime of the texture bags approach described in Chapter 3.

One difference between the approaches ERSP and RLSV is the splitting function (Equation 4.11 and Equation 4.12). Another difference is the issue of vocabulary size. Due to the binary splits, Random vocabularies are growing exponentially [52]. Pauly et al. does not address this issue as they use the BoVW vectors for classification rather than retrieval. For several reasons (e.g. Memory consumption and search time) smaller vocabulary size is desired in CBIR [52]. Mu et al. introduced a stop list to address this problem. In addition, they propose as a random vocabulary method which allows to use labeled training data in order to generate a more discriminative vocabulary.

The random partitioning of the feature space allows to analyze the distribution of feature associated information like tags or labels and thus can be used to find potential relationships between feature components and labeling. The classification solution described in Section 4.1 is an example of such an approach.

In this thesis, Random Ferns are used for vocabulary building, weakly supervised learning and to generate a hamming embedding for indexing.

CHAPTER 5

State of the Art: Vector Indexing

The calculation of the Euclidean distance between vectors is used in applications such as classification [70], clustering [18] and the retrieval of similar images [15, 41, 4]. This chapter examines state of the art methods for searching a collection of vectors for nearest neighbors to a query vector in an efficient way.

The first section will give a short formal problem definition and a notation for this chapter. Section 5.2 presents methods for finding the exact nearest neighbors and Section 5.3 examines and discusses approximate nearest neighbor search methods which sacrifice a certain amount of accuracy for higher performance in retrieval speed and memory consumption. Finally, Section 5.4 discusses the relevance of the examined methods for this thesis.

5.1 Problem definition and Notation

In this chapter, we focus on the nearest neighbor search in a d-dimensional Euclidean space $\mathbb{R}^d$. Let our finite set of N records be defined as a set of vectors $\mathcal{N} \subset \mathbb{R}^d$ given by $\mathcal{N} = \{\mathbf{n}_1\ \mathbf{n}_2\ ...\ \mathbf{n}_N\}$. There are different query types an indexing mechanism can be optimized and suited for [15], e.g. range or point queries. In this work only methods for nearest neighbor point queries in metric spaces are considered. Lets define the nearest-neighbor search problem (NN) for a query record $\mathbf{q} \in \mathbb{R}^d$ by finding the element $NN(\mathbf{q}, \mathcal{N})$ in the set $\mathcal{N}$ where:

$$NN(\mathbf{q}, \mathcal{N}) = \underset{\mathbf{n} \in \mathcal{N}}{\operatorname{argmin}}\ d_{Euc}(\mathbf{q}, \mathbf{n}) \tag{5.1}$$

and $d_{Euc}(\mathbf{q}, \mathbf{n})$ denotes the Euclidean distance between $\mathbf{q}$ and $\mathbf{n}$. Furthermore, we define the k-nearest-neighbors problem (kNN) as finding the set of k records with the smallest Euclidean distance to the query point $\mathbf{q}$:

$$kNN(\mathbf{q}, k, \mathcal{N}) = \{NN(\mathbf{q}, \mathcal{N}), NN(\mathbf{q}, \mathcal{N} \setminus NN(\mathbf{q}, \mathcal{N})), \dots\} \tag{5.2}$$

There are three main objectives considered in this work that an indexing method can address:

1. **Speed**: The indexing method should allow to find the k-nearest-neighbors faster than the straight forward approach if calculating the Euclidean distance between the query vector and all records in $\mathcal{N}$.

2. **Accuracy**: If the method is approximating the nearest neighbors, the quality of the approximation is of importance.

3. **Memory consumption**: The indexing method should address the issue of memory consumption per record (e.g sacrificing accuracy for memory consumption)

Note, that not all indexing methods address all of the stated objectives so that the decision for a certain method may already desire a trade off regarding these properties.

5.2 Hierarchical Indexing Structures

The operation of the nearest neighbor search can be achieved by searching a list built from $\mathcal{N}$. A trivial nearest neighbor search algorithm would scan the entire list and compare $\mathbf{q}$ to each $\mathbf{n}$ in $\mathcal{N}$ (*linear scan*). This algorithm has the time complexity of $\theta(N)$. By structuring the vector-set more intelligently, it is possible to avoid making a distance computation for every member [50]. Different tree-based indexing structures suitable for nearest neighbor search in higher dimensions are described by Bohm et al. [15]. The methods are differently affected by factors like dimension, database size and structure in data [15] and also have varying properties like suitability to use secondary memory or computing time for construction, searching, deleting and inserting [15]. This section will examine two state of the art methods for such a data-structure to give an idea how the problem of nearest neighbor search can be solved extending search strategies for lower dimensional spaces. The kd-tree is examined in Section 5.2.1. This indexing method extends the idea of a binary search tree by iteratively partitioning the whole data space. In Section 5.2.2, a different tree based indexing structure called the R-tree is examined. R-tree follows the paradigm of hierarchical clustering by building minimum bounding boxes around sub-sets of $\mathcal{N}$.

5.2.1 The kd-tree structure

The kd-tree was originally introduced and later optimized and examined in detail by J.L. Bentley et al. [29, 9]. It is a searchable data structure for storing a finite set of points in a d-dimensional space. The kd-tree structure provides a mechanism for examining only those records which are close to the query record and reduces the computation time required to find the best matches.

Space partitioning

Bentley et al. defined the kd-tree is a generalization of a binary search tree [29]. Each node holds a subset of the whole set of records and a partitioning of it. The root, as a special case represents the entire set. Each none terminal node has two child nodes which represent the two subsets defined by the partitioning. The terminal nodes represent mutually exclusive subsets of

Field Name	Field Type	Description
$\mathbf{n}_n$	k-dimensional vector	a vector, holding the keys of record n
j_n	integer	discriminator, $1 <= DI_n <= k$
LC_n	kd-tree	a kd-tree representing the records left of this node
RC_n	kd-tree	a kd-tree representing the records right of this node

Table 5.1: The fields of a kd-tree node n. $\mathbf{n}_n$ is the splitting-point. LC_n and RC_n are sub-trees representing the nodes on either side of a hyperplane splitting the space perpendicular to the direction of the j_n field.

the record space. Thus, all terminal nodes collectively form a partitioning of the whole record space.

A node in a kd-tree is basically a split of one dimension of the record space. This split is defined by the splitting dimension j_n called *discriminator* and a record $\mathbf{n}_n \in \mathcal{N}$. The split is performed along the hyperplane induced by $\mathbf{n}_n^{[j_n]}$ called the discriminator value denoting the j_n dimension of vector $\mathbf{n}$. The two partitions induced by the split can be denoted by LC_n and RC_n so that

$$\mathbf{l}^{[j_n]} < \mathbf{n}_n^{[j_n]} \wedge \mathbf{r}^{[j_n]} \geq \mathbf{n}_n^{[j_n]} : \forall \mathbf{l} \in LC_n \wedge \forall \mathbf{r} \in RC_n. \tag{5.3}$$

Table 5.1 gives an overview of the fields of a kd-tree node. Note, that in this definition of a kd-tree, every record is stored in a splitting-node and the terminal nodes are all empty [9].

In the one dimensional case, a kd-tree is a normal binary search tree with the record-values as keys. However, in the d dimensional case, one record is represented by d keys and every one of these can serve as a discriminator for partitioning the subset represented by a particular node in the tree [29]. There are different approaches of which the key gets chosen to be the discriminator for a node. The original algorithm of Bentley [9] iterates through keys in order on the basis of nodes level in the tree and the partition values are random key values in each subset. An optimized version of the kd-tree structure selects the key with the largest spread in values as discriminator and chooses the median as the partition value [9]. Figure 5.1 gives a 2-dimensional example of a basic kd-tree. Note, that in this example the tree is build upon the last level and the terminal nodes are empty like proposed in [9]. However, it has been shown, that for nearest neighbor queries and larger record sets, non-empty leaf nodes are more efficient [29].

Search algorithm

One can see, that a nodes position in the tree is bound by the discriminator values of all superior split nodes. The search algorithm is a recursive procedure, starting at the root of the tree. At each node, the geometric boundaries that delimit the two subsequent subsets can be computed. Due to the nature of search trees, the volumes which contain subsets defined by nodes deeper in the tree are smaller than those on higher levels [29].

In the following the search algorithm presented in [29] is summarized. Note, that this search algorithm is suited for trees with non-empty leaf nodes. A query vector $\mathbf{q}$ can descend the tree by comparing its discriminator value $\mathbf{q}^{[j_n]}$ to the value of the split node $\mathbf{n}_n^{[j_n]}$ and depending on the result proceeding with LC_n or RC_n. If the investigated node is a terminal node, all records in

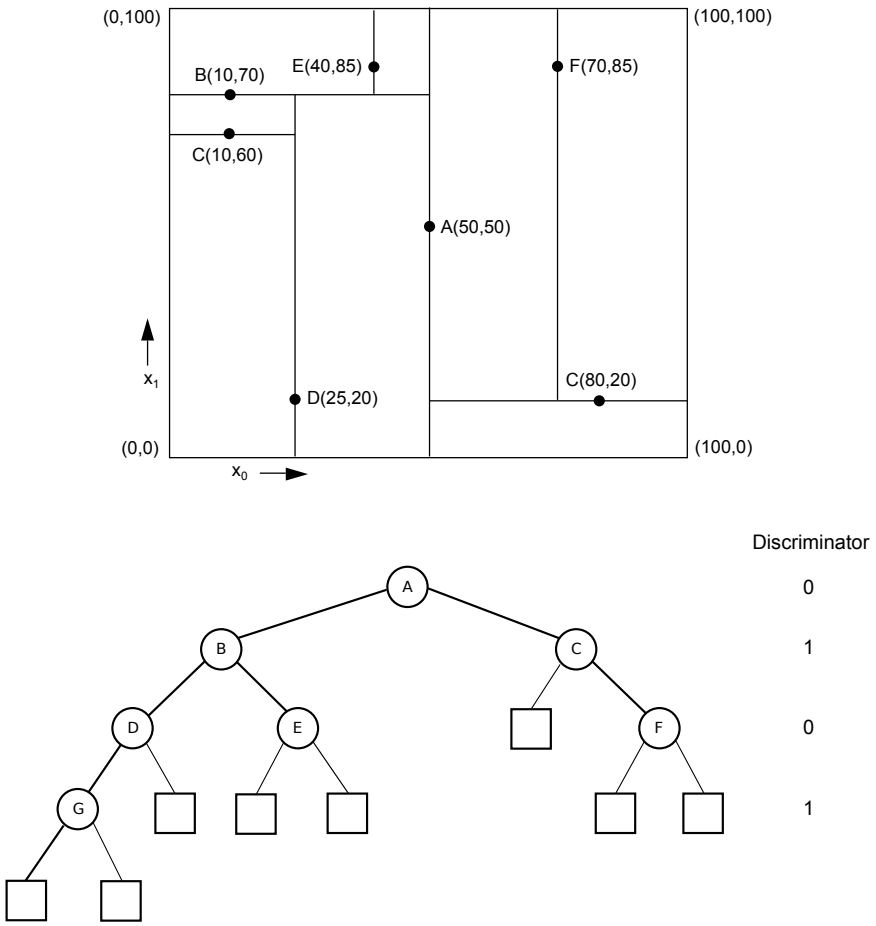


Figure 5.1: A 2-dimensional kd-tree like it was originally proposed in [9]. At the top, records of a 2-dimensional feature space are stored in nodes. The boxes represent the range of a subtree. At the bottom, the graph representation is depicted, were left childs are nodes with a lower value for the nodes discriminator dimension. Right childs are subtrees with higher discriminator values. Boxes are empty child nodes. (Figure according to [9])

the remaining bucket are examined. For the search, a list $\mathcal{C}$ of the m closest records encountered at a time and their distance to the query vector is maintained. Once, a terminal node has been examined, a test is made to decide weather the opposite partition has to be considered to hold one of the m closest neighbors or not. That is the case, if a ball centered at the query record with radius r of the distance of the query $\mathbf{q}$ to $\mathcal{C}_m$ overlaps the geometric boundary which delimit the records in the parent node. This test is called "bounds-overlap-ball test" [29]. If the test fails another test is made to decide weather the search is over or one more parent node has to be examined. This test is called "ball-within-bounds test"[29] and checks if the ball fully lies within the bounds of the current partition.

5.2.2 The R-tree structure

The R-tree indexing structure was originally proposed by Antonin Guttman [32] for indexing spacial objects like polygons. Though, it was also used for indexing point objects [15]. The R*-tree [6], R+-tree [68] and the X-tree [11] are extensions of the R-tree to optimize the structure for point objects and for higher dimensional spaces.

Tree Structure

The R-tree is a balanced search tree which consists of two different kind of nodes, the *directory nodes* and the *data nodes*. All leaf nodes of the tree are data nodes. The indexing records are only stored in the leafs of the tree. The directory nodes consist of (key,pointer)-tuples where the keys are bounding boxes and the pointer point to another directory or data node [32]. In contrast to the kd-tree, the R-tree does not partition the whole feature space but quantizes into rectilinear Minimum Bounding Rectangles (MBR). MBRs are axis-parallel multidimensional rectangles which are minimum approximations of an enclosed point set, so called page regions, where every surface area contains at least one data point [15]. A MBR I can be represented as sets of upper and lower bounds representing multidimensional intervals for each dimension [32] with

$$I = [lb_1, ub_0] \times ... \times [lb_d, ub_d]. \tag{5.4}$$

The space partitioning is neither complete nor disjoint, so that some empty parts of the data space may be not covered at all but overlapping between regions is allowed [15].

Let I be the MBR of a node in an R-tree and M be a parameter which defines the maximum number of entries for one node and $m \leq M/2$. Guttman stated following properties for an R-tree:

- Every leaf node contains between m and M index records unless it is the root.
- For all index records in a leaf node, I is the smallest rectangle that spatially contains the set of records.
- Every non-leaf node has between m and M children unless it is the root.
- For each entry in a non leaf-node, I is the smallest rectangle that spatially contains the rectangles in the child nodes.
- The root node has at least two children unless it is a leaf.
- All leaves appear on the same level, thus the tree is balanced.

For the inserting procedure of a new point, a suitable page has to be searched. Three different cases my occur during the search [15]:

- If the point is contained in exactly one page region, the corresponding page is used.

- If the point is contained in several different page regions, the page region with the smallest volume is used.
- If no region contains the point, the region is chosen, which yields the smallest enlargement. If more than one region show smallest enlargement, the one with the smallest volume among them is chosen.

By applying these rules, points are inserted to page regions. When one page would overflow, so that it contains more than M entries, the page is spit. Figure 5.2 shows a two dimensional R-tree with 3 levels and $M = 3$.

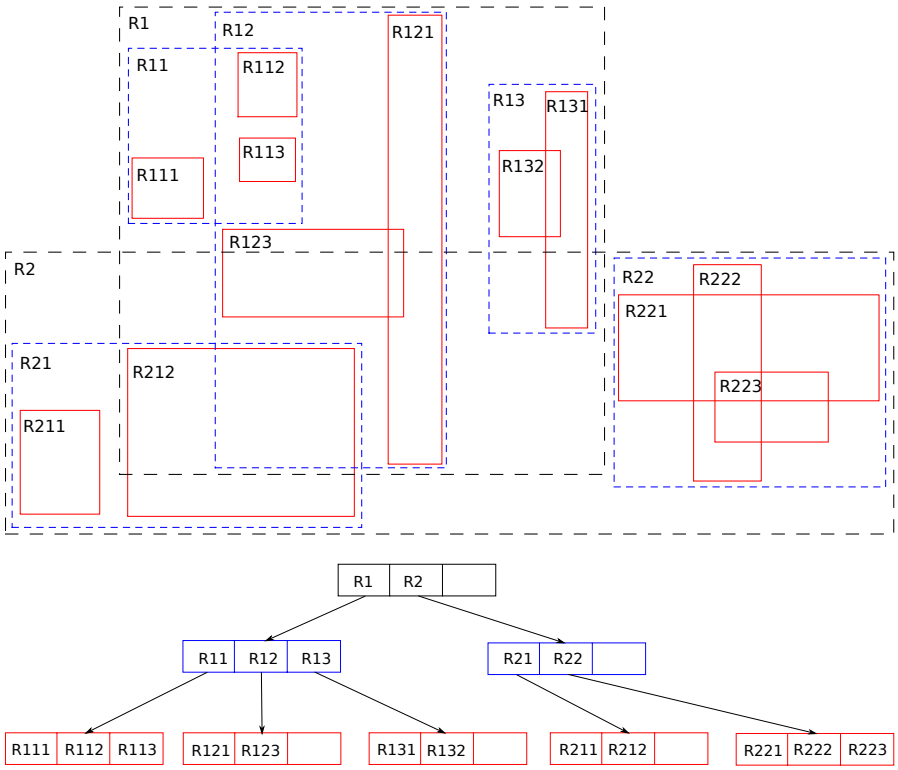


Figure 5.2: An R-tree of order 3 as proposed by Antonin Guttman. The top half of the figure shows the space partitioning by MBRs (R1-R19) on three levels. The lower half depicts the tree representation. The leaf nodes (red) hold the indexed records. (Figure according to [32])

Search algorithm

There are more than just one algorithm to process nearest neighbor queries in R-tree structures [15]. Due to its ability to process point and k-NN queries ,the algorithm of Hjaltason and Samet [35] is described only. The algorithm manages an Active Page List (APL). In the APL all pages are marked active if its parent has been processed but not the page itself. As the parent has been loaded already, the regions of all the active pages are known and the minimum distances of the

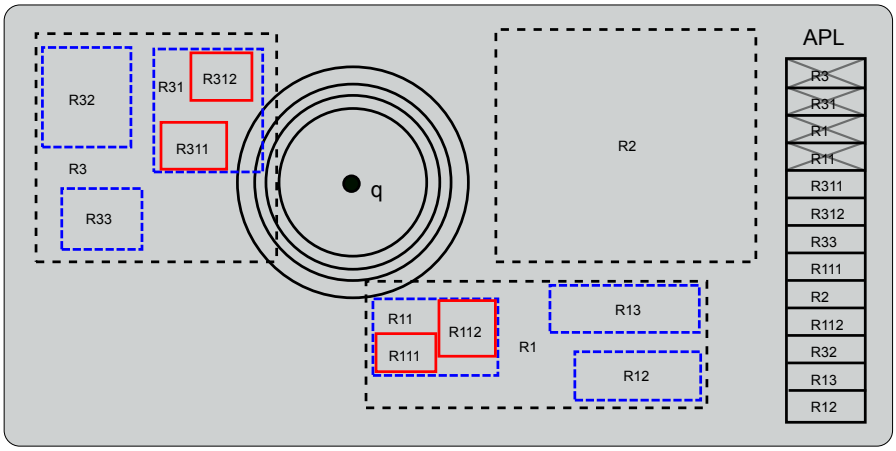


Figure 5.3: The nearest neighbor search algorithm of Hjaltason and Samet [35]. The pages are accessed not in order induced by hierarchy but in order of increasing distance to the query point. (Figure according to [15])

query to the regions can be determined. Let $\mathbf{q}$ be a query point and I the MBR of a region, the minimum distance (MDIST) is defined as the distance of $\mathbf{q}$ to an edge or a corner of I [15]:

$$MDIST^2(q, I) = \sum_{i=1}^{d} \left(\begin{cases} lb_i - \mathbf{q}^{[i]} & \text{if } \mathbf{q}^{[i]} < lb_i \\ 0 & \text{otherwise} \\ \mathbf{q}^{[i]} - ub_i & \text{if } ub_i < \mathbf{q}^{[i]} \end{cases} \right)^2 \tag{5.5}$$

Like in the kd-tree search algorithm, a priority queue $\mathcal{C}$ of the m closest records is maintained when a k-NN search is performed. Bohm et al. [15] interpreted and summarized the iterative processing step of the algorithm like in the following:

1. Selecting the page p from the APL with the minimum distance to $\mathbf{q}$
2. Loading p
3. Deleting p from the APL
4. If p is a data page, compare and update the priority queue $\mathcal{C}$ with the points in p

These steps are repeated until the point on position m in $\mathcal{C}$ is closer than the nearest active page.

5.2.3 Curse of Dimensionality

The term "*Curse of Dimensionality*" was coined by Richard E. Bellman and is used to refer to various negative effects occurring when the number of dimensions increases [8]. In this section, phenomenons which effect indexing methods only, are discussed.

Some effects are caused by the fact that important parameters like volume and surface depend exponentially from the dimensionality of the data space. Bohm et al. [15] use an interesting lemma to demonstrate, that we can not extrapolate our knowledge of lower $(1-3)$ to higher (>3) dimensional spaces:

Considering a $[0,1]^d$ space with the center-point c at $(0.5,...,0.5)$. The lemma "Every d-dimensional sphere touching (or intersecting) the $(d-1)$-dimensional boundaries of the data space also contains c." [15] is true for $d=2$ and also $d=3$. For higher dimensions like $d=16$ a counter example can be shown: A sphere around the point $p=(0.3,...,0.3)$ of radius 0.7 obviously touches or intersects all 15-dimensional surfaces of the data space. However, the Euclidean distance to c is $\sqrt{16 \cdot 0.2^2} = 0,8$ and thus, not covered by the hypersphere.

Another relevant effect is the behavior of the volume of the unit hypersphere with increasing dimensionality. In high-dimensional spaces it is increasingly inaccurate to approximate a hypersphere by a hypercube. The unit-hypersphere is the higher dimensional equivalent to the unit-circle and unit-sphere in higher dimensions ($d \geq 4$), thus a hypersphere with unit-radius 1. By assuming an increasing number of dimensions the volume of the hypershpere will at first increase and after dimension 6 decrease and converge to 0 [78]. Figure 5.4 shows the volume of a hypersphere with respect to the dimensionality.

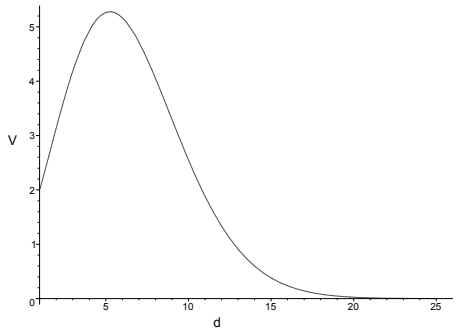


Figure 5.4: The volume of a unit hypersphere with respect to the dimensionality. While the volume increases at first, it decreases from dimension 6 and converges to 0 from about dimension 20.

In a further step, one can think about a bounding-box enclosing the hypersphere. It may be intuitive to think, that the volume of the bounding box will correlate with the one of the enclosed sphere. However, the exact opposite is happening and the sphere/cube volume ratio converges to 0 [78].

The indexing structures examined are based on space partitioning by means of $(d-1)$-dimensional hyperplanes. At a node in a kd-tree for instance, all data items with a value smaller than the split value in one dimension are assigned to the first, whereas the others form the second partition. This splitting is continued until a certain threshold for a number of entries in the leaf nodes is reached. The height of the tree is logarithmically depending on the number of leaf nodes. Lets consider a kd-tree with the average number of d' splits to that

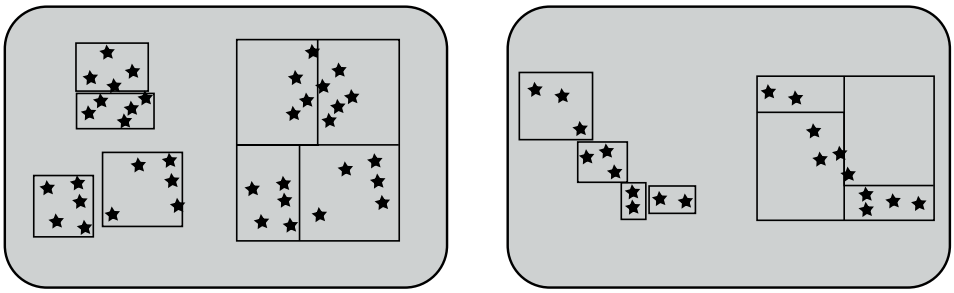
$$d' = log_2(\frac{N}{L}), \tag{5.6}$$

where L is the capacity of a single leaf node. When considering a 20-dimensional data space at least 2^{20} data points would need to be indexed to achieve just one split per dimension [15]. That means, that the partitions will be located at the border of the data space and on higher dimensions will span the complete space on most directions.

The effects described are responsible for decreasing performance of tree based indexing methods by increasing dimensionality [80]. The Texture Bag approach (Chapter 3) generates feature vectors of dimensionality 1200. The X-tree, which is an R-tree that has been specially trimmed for higher dimensions degenerates to a linear scan at about 20 dimensions [80]. To the best of our knowledge, there is no method that allows to determine the exact nearest neighbor in sublinear time for dimensions higher than 20.

5.2.4 Discussion and Comparison

Each non-terminal node of the kd-tree has to store the discriminating key number and the partition value. The memory footprint of one record is therefore increased by these two values. The R-tree has to store one MBR in every node, which size directly depends on the dimensionality of the data. In contrast to the kd-tree, the records are not stored in the nodes. The kd-tree and the R-tree are search trees, thus the time required to traverse the tree from node to leaf is logarithmic in the number of leaf nodes, which is directly proportional to dataset size N. However, the kd-tree is binary and therefore in log_2 while the R-tree is in log_M. Evaluations of the kd-tree show, that for the case of uniformly distributed data, the backtracking in the tree during the search is constant and independent to N and the expected search time to find the kNN to a specified query record q is in $\Theta(\log(N))$ [29]. Theoretically, the R-tree shows the same performance. However, for the case of structured data, the kd-tree will have to access more nodes due to its complete space decomposition [15]. In comparison, the R-tree covers just a part of the feature space. For correlated or clustered data, this way of space decomposition is superior to complete partitioning because the page regions are smaller and yield a lower probability to be accessed during search [15]. Figure 5.5 illustrates the differences between the two ways of space decomposition. It has



(a) (b)

Figure 5.5: Incomplete (R-tree) vs. Complete (kd-tree) space decomposition on clustered (a) and correlated (b) data (Figures according to [15])

to be considered, that not only the distribution of the data but also the number of data items and the dimensionality have an influence on the performance of an index structure [15].

5.3 Approximate Nearest Neighbor Search

The curse of dimensionality restricts the use of exact sublinear NN indexing methods like the kd-tree or the R-tree to lower and medium dimensional spaces up to about 20 dimensions [80, 12, 10]. To avoid a linear scan through all vectors of a database, methods solving the Approximate Nearest Neighbor (ANN) search problem are proposed. The key idea of ANN is to find the nearest neighbors with high probability, instead of finding the exact nearest neighbors with probability 1 by sacrificing a certain amount of accuracy for higher performance in retrieval speed and memory consumption compared to exact methods [41].

There are different approaches to accomplish that task which we can divide into two groups. The first group generates candidate lists based on an approximate search for a query vector. Subsequently, this result-list is resorted based on the exact distance of the vectors. The second group of methods omits the second step of reordering and relies fully on the comparison of compact codes (also denoted as sketches) to estimate the distance of the original vectors. This has the advantage, that the original vectors do not have to be stored and managed. Early examples for the first group of methods are Vector Approximation Files (VA-files) [56] and different variations of Local Sensitive Hashing (LSH) [39, 31, 3]. The Fast Library for Approximate Nearest Neighbors (FLANN)[1] [53] is a collection of approximate nearest neighbor search techniques. It has been shown, that this library outperforms VA-files and LSH, thus, this library will be examined in more detail in Section 5.3.1. The FLANN library utilizes hierarchical k-means clustering and multiple randomized kd-trees but also comes with a LSH implementation. In addition, it provides automatic algorithm and parameter selection in order to adopt to the target data.

Two state of the art methods for the latter group are Spectral Hashing [81] and Product Quantization [41]. Spectral Hashing compresses the vectors to binary codes utilizing spectral techniques were the Hamming distance of the binary codes correlates with the Euclidean distance of the original feature vectors. Product Quantization generates short lists of integer values and lookup tables from which an approximation of the Euclidean distance in the original space can be calculated. Product Quantization is examined in Section 5.3.4 and Spectral Hashing in Section 5.3.2. These two methods are compared and discussed in Section 5.3.5.

5.3.1 FLANN

Muja and Lowe presented a library for finding approximate nearest neighbors [53] utilizing a modified version of the hierarchical k-means trees [30, 56] and the multiple randomized kd-trees method of Anan and Hartley [72]. The FLANN library was integrated into the OpenCV[2] framework and forms the central component which provides the functionality of approximate nearest neighbor search [59].

Hierarchical k-means Tree

The hierarchical k-means tree is built by splitting the data points into k distinct partitions using the k-means clustering algorithm and recursively repeating the splitting for each partition [30].

[1] http://www.cs.ubc.ca/research/flann/
[2] http://opencv.org/

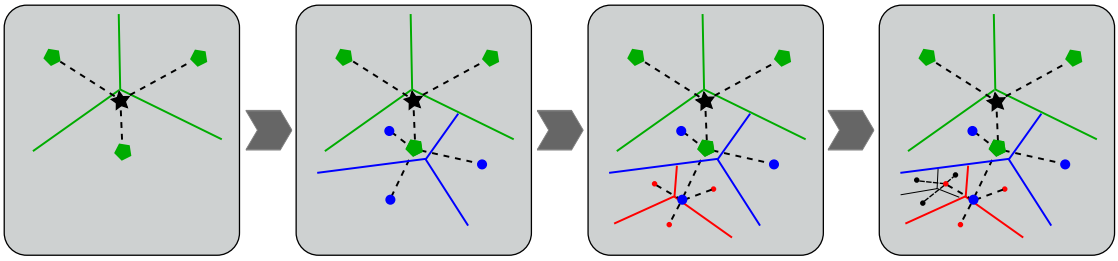


Figure 5.6: An illustration of the hierarchical k-means clustering algorithm. In this case, the branching factor is $k = 3$. (Figure according to [56])

Thus, k defines the branching factor of the tree. The k-means process is applied on a training subset, defining k cluster center. The k-means clustering partitions the feature space into k voronoi cells, where each point is assigned to the closest centroid. This process is recursively repeated level by level for each new partition, up to a defined maximum level L [30] (Figure 5.6).

In the online phase, each data point is propagated down the tree by comparing the vector to the k candidate cluster centers and and the closest one is chosen. That means that a point can be indexed by $k \cdot L$ distance computations. At the end, a hierarchical k-means tree splits the feature space into k^L mutually exclusive partitions. Following the inverted file paradigm, a leaf node of the tree contains a list of identifier of the points which landed in the partition.

Muja and Lowe developed an algorithm which explores the tree in a best-bin-first manner [53]. While propagating down the tree for a query point, a priority queue is maintained, holding all unexplored centroids in each node along the path. When a leaf node is reached, the points, stored in the leaf are added to a candidate list and the search is restarted from the closest center stored in the priority queue. The level of approximation is defined by a predetermined number of leaf nodes which are examined. In a last step, the points stored in the candidate list are sorted according to their distance to the query point.

Multiple Randomized kd-trees

The general idea behind the kd-tree has already been described in Section 5.2. It's characteristic of degrading search performance was also examined. This problem may be overcome by terminating the search after a specified number of nodes are searched, resulting in an approximate result [7]. The increasing number of searched nodes increases the probability of finding the true nearest neighbor [7]. However, any newly examined cell depends on the previously examined cells and the search of successive nodes on one tree are not independent [72]. Anan and Hartley proposed the idea of multiple search-trees to enhance independence by building multiple kd-trees with different parameters like selection of the partitioning value or the dimension [72]. They proposed several different approaches to enhance independence of the trees:

- NKD-tree. By randomly rotating the dataset for each tree, different sets of dimensions are covered and the resulting kd-trees have a different structure.

- RKD-tree. The RKD-tree achieves independence by randomly selecting the splitting dimension among a set of dimensions with highest variance instead of selecting the one with highest variance always.

- PKD-tree. Like the NKD-tree, but the rotation is preformed to align the axis of the principal components of the dataset.

Comparable performance for each of the methods have been showed but one method may be better suited depending on the dataset [72]. However, FLANN implements the RDK-tree [53].

The search in the multiple trees is performed simultaneously using a pooled priority queue. After each tree has been descended by the query to find an initial nearest neighbor candidate, the closest of the candidates found is used. As the priority queue is pooled, the nodes are ranked independent of their tree membership.

5.3.2 Spectral Hashing

Spectral Hashing was introduced by Weiss et al. in [81]. The basic idea is to produce binary codes of feature vectors so that the Hamming distance between the codes reflects the Euclidean distance of the original vectors. This has two advantages (1) the calculation of the Hamming distance of two binary codes is faster than the calculation of the Euclidean distance and (2) the binary codes need less memory than the original vector.

Spectral Hashing is not the only method which finds a mapping function that transforms each object into a binary code, where items that are close in Euclidean space are mapped to similar binary codewords. For a short survey look at [81]. However, Spectral Hashing was chosen for this chapter due to its promising results presented in [81].

Weiss et al. stated some criteria a good binary code should meet [81]:

- be easily computed for a novel input,

- require a small number of bits to code the full dataset,

- map similar items to similar binary codewords.

In respect to the latter two requirements, a code is efficient if each bit has a 50% chance of being one or zero and that the bits are independent of each other [81].

Spectral Relaxation

Spectral Hashing reduces the problem of creating good binary codewords to a particular form of graph partitioning. To solve the graph partitioning problem, an approach related to spectral algorithms like Spectral Clustering and Laplacian Eigenmaps is suggested.

In particular, N datapoints can be seen as vertices in an undirected graph and the weight between two vertices $\mathbf{n}_i$ and $\mathbf{n}_j$ is given by an affinity matrix $\mathbf{W}_{N\times N}$. Where

$$\mathbf{W}(i,j) = e^{(-\|\mathbf{n}_i - \mathbf{n}_j\|^2/\epsilon^2)} \ . \tag{5.7}$$

Considering Equation (5.7) the similarity correlates with the Euclidean distance and the parameter ϵ defines the distance in R^d which corresponds to items that are assumed to be similar.

In the next step a diagonal matrix $D_{n\times n}$ is introduced where

$$\mathbf{D}(i,i) = \sum_{j=1}^{N} \mathbf{W}(i,j) \ . \tag{5.8}$$

The graph Laplacian $\mathbf{L}$ is build by $\mathbf{D} - \mathbf{W}$. The s bits are calculated by thresholding the first k eigenvectors with the smallest eigenvalues of the Laplacian except the eigenvector with eigenvalue 0.

Hamming embedding

A Hamming Embedding (HE) is a mapping from the Euclidean space into the Hamming space that ensures that the Hamming distance h between a point and its nearest neighbors in the Euclidean correlates [40]. More formally, in a HE, a vector $\mathbf{n}$ is encoded into a d_b-dimensional binary signature $b(\mathbf{n}) = (b_1(\mathbf{n}), ..., b_{d_b}(\mathbf{n}))$. The mapping function is designed so that the Hamming distance

$$d_{Ham}(b(\mathbf{q}), b(\mathbf{n})) = \sum_{1\leq i\leq d_b} 1 - \delta_{b_i(\mathbf{q}),b_i(\mathbf{n})} \tag{5.9}$$

reflects the Euclidean distance $d_{Euc}(\mathbf{q}, \mathbf{n})$ of two vectors $\mathbf{q}$ and $\mathbf{n}$ [40].

To represent feature vectors as binary codes not only allows to reduce the memory footprint of a record, but also facilitates fast searching as instead of calculating the Euclidean distance just bitcount operations are needed to retrieve similarity. In [67], searching through 1 million 128-bit codes takes 3.6 milliseconds utilizing a C implementation on a 3GHz Intel Xeon processor. Furthermore, they propose a method which treats binary codes of 30 bits as memory addresses and hence allows to retrieve candidates lists (lists of records closer than a certain Hamming distance) without any search in constant time.

5.3.3 Binary Codes by LSH for Approximate Euclidean Distance

The Hamming distance of SH codes reflects the Euclidean distance between the corresponding original vectors. However, the actual Euclidean distance is not approximated and will remain unknown, only the Hamming distance can be used to judge similarity. Murakatat et al. present an approach, to approximate the Euclidean distance from binary codes generated by random projections [48] posing a Locality Sensitive Hash Function (LSH).

Binary Codes by LSH Functions

The binary code generation is performed by a member of the LSH family. LSH is a group of hashing functions, that are locality sensitive (e.g. the probability of outputting the same code of close objects is high and that of distant objects is low [39]). One group of LSH uses hash functions of the form [22]:

$$h(\mathbf{n}) = sign(\mathbf{u}^\intercal \mathbf{n}), \tag{5.10}$$

where $\mathbf{u}$ is a random unit-length vector. It is worth noting, that this function corresponds to the RF split function of Mu et al. (Equation 4.11). For this function, the probability of collision is [22]:

$$Pr[h(\mathbf{q}) = h(\mathbf{n})] = 1 - \frac{\theta(\mathbf{q}, \mathbf{n})}{\pi}. \tag{5.11}$$

From Hamming to Approximated Euclidean Distance

The collision probability (Equation 5.11) varies proportional to the cosine of $\theta(\mathbf{q}, \mathbf{n})$ [22]. The probability can be estimated directly from the number m of random projections performed as it corresponds to the Hamming distance of the binary codes obtained by the hash functions [48]:

$$\hat{\theta}(\mathbf{q}, \mathbf{n}) = \frac{\pi}{m} d_{Ham}(b(\mathbf{q}), b(\mathbf{n})) \tag{5.12}$$

where $b(.)$ is the binary encoding function consisting of a set of LSH hash functions.

The Euclidean distance between two vectors can be expressed by

$$\begin{aligned} d_{Euc}(\mathbf{q}, \mathbf{n})^2 &= \|\mathbf{q} - \mathbf{n}\|^2 &&= \|\mathbf{q}\|^2 + \|\mathbf{n}\|^2 - 2\mathbf{q}^\intercal \mathbf{n}, \\ &&&= \|\mathbf{q}\|^2 + \|\mathbf{n}\|^2 - 2\cos(\theta(\mathbf{q}, \mathbf{n})) \end{aligned} \tag{5.13}$$

In combination with Equation 5.12, the Euclidean distance can be approximated by following equation [48]:

$$\hat{d}_{Euc}(\mathbf{q}, \mathbf{n})^2 = \|\mathbf{q}\|^2 + \|\mathbf{n}\|^2 - 2\cos(\frac{\pi}{m} d_{Ham}(b(\mathbf{q}), b(\mathbf{n}))) \tag{5.14}$$

In the following, this approach is called Locality Sensitive Hashing for Euclidean (LSHE) distance approximation.

5.3.4 Product Quantization

Product Quantization (PQ) is an encoding technique proposed by Jégou et al. [41]. In difference to the former presented methods SH and LSHE, the codes are not binary but integer values which represent quantizer centroids in subspaces.

Quantization

In the following, a short introduction of quantization is given based on the definition of Jégou et al. [41].

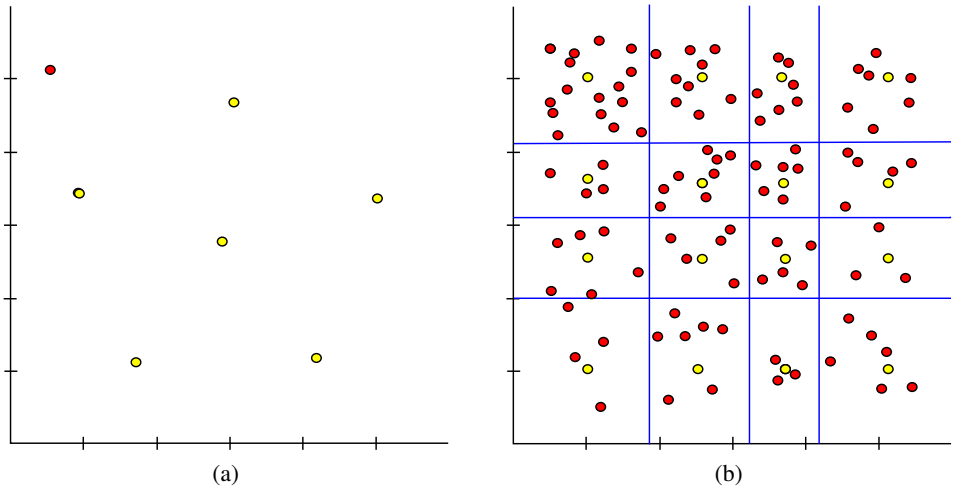


Figure 5.7: Illustration of the a regular quantizer (a) on all dimensions and a product quantizer (b) on orthogonal subspaces.

Quantization is a method to reduce the cardinality of the representation space. A quantizer can be seen as a function $q : \mathbb{R}^d \rightarrow \mathbb{R}^d$ mapping a d-dimensional vector $\mathbf{n}$ to a vector $q(\mathbf{n})$. Where $q(\mathbf{n})$ is a reproduction value (or centroid) out of a finite set $\mathcal{C}$ called codebook. All vectors mapped to a given centroid form a Voronoi cell and all C cells of a quantizer form a partition of $\mathbb{R}^d$.

Quantization with Subquantizer

Producing high numbers of partitions is very hard or even impossible [41]. In contrast to ordinary quantization, product quantization does not partition the whole space but orthogonal subspaces. Figure 5.7 illustrates the difference between a regular quantizer and a product quantizer. The quantizers of subspaces are called *subquantizer*. When using m subspaces, $\mathcal{C}_1 ... \mathcal{C}_m$ different codebooks are generated. The overall codebook finally is defined as the cartesian product of the codebooks

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times ... \times \mathcal{C}_m \ . \tag{5.15}$$

The centroids of this codebook are the concatenations of the centroids of the subquantizers. If we assume, that each subquantizer generates the same number of controids C^*, the total number of centroids is $C = (C^*)^m$. Thus, with relatively small numbers of C^* a large set of centroids can be produced. To make that possible, an input vector $\mathbf{n}$ is split into m distinct subvectors $u_j(\mathbf{n})$, $1 \leq j \leq m$ of dimension $d^* = d/m$. By using m distinct quantizers, these subvectors are quantized separately. It should be considered, that saving the final centroids of $\mathcal{C}$ is not efficient, so that only the centroids of the subquantizers are saved.

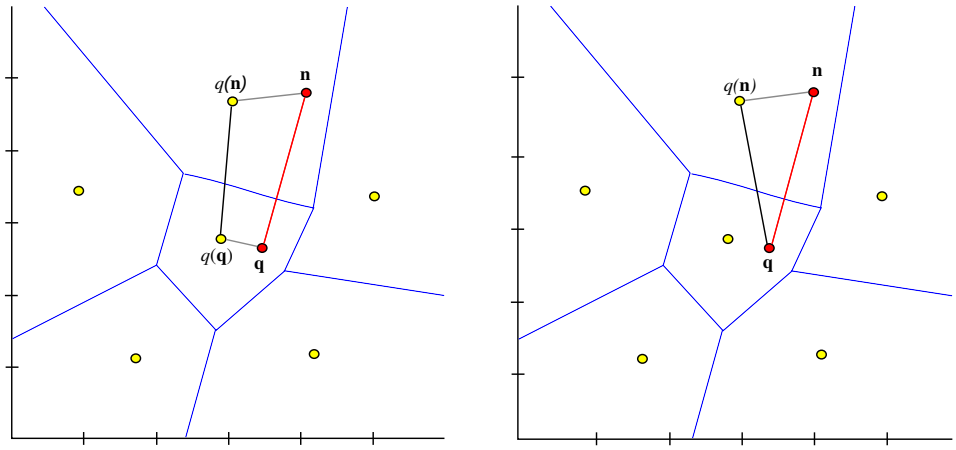


Figure 5.8: The symmetric and the asymmetric distance calculation. In the symmetric case (a), the distance is computed as the distance of the centroids while in the asymmetric way (b) the query vector is not encoded. (Figure according to [41])

Computing Distance

Jégou et al. propose two methods for distance computation of a query vector to the quantized codes: A symmetric and an asymmetric way. In the symmetric way, the query vector is encoded by the subquantizers, and the squared distances to the centroids of each database code is summed up. In the asymmetric way, the query vector is not encoded. Figure 5.8 shows an illustration of the two methods. Jégou et al. state, that the only advantage of the symmetric way is, that the query vectors need less memory when they are encoded, and that is in most cases negligible [41].

In the following, the Asymmetric Distance Computation (ADC) is described. As mentioned above, the query $\mathbf{q}$ is not encoded, while a database vector $\mathbf{n}$ is represented by its quantizer indices. The Euclidean distance $d(\mathbf{q}, \mathbf{n})$ is approximated by the distance between the query point and the centroid of the database vector such that

$$\tilde{d}(\mathbf{q}, \mathbf{n}) = d(\mathbf{q}, q(\mathbf{n})) = \sqrt{\sum_j d(u_j(\mathbf{q}), q_j(u_j(\mathbf{n})))^2} \tag{5.16}$$

The squared distances $d(u_j(\mathbf{q}), \mathbf{c}_{j,i})^2$ for $j = 1...m$ and $i = 1...C^*$ are computed in advance so that the distance computation for one vector is an add up of m values. The calculation of the square root is omitted as the square root function increases monotonically and does not influence the vector ranking.

Non-exhaustive Search

Jegou et al. showed, that the distance computation based on the PQ-codes is faster compared to a calculation of the Euclidean distance of the original vectors [41]. However, the proposed

method is still exhaustive search and the complexity is in $\theta(N)$. To avoid exhaustive search, Jegou et al. combined an inverted file structure with the asymmetric distance computation which he calls IVFADC [41]. Like in the hierarchical k-means tree structure described in Section 5.3.1, the index entries are quantized and the corresponding vector identifiers are stored in the lists. However, in difference to the indexing structure the identifier are stored along with a PQ-code and IVFADC uses non-hierarchical k-means clustering with C'' cluster centers. This first quantization step was termed "*coarse quantizer*" [41]. The PQ-code assigned to a centroid encode the difference between the vector and the centroid of the coarse quantizer. More formally, the offset from the vector $\mathbf{n}$ to the coarse quantizers centroid $q_c(\mathbf{n})$ is the residual vector

$$r(\mathbf{n}) = \mathbf{n} - q_c(\mathbf{n}). \tag{5.17}$$

A product quantizer q_p is used to encode $r(\mathbf{n})$, resulting in the vector approximation

$$\ddot{\mathbf{n}} \triangleq q_c(\mathbf{n}) + q_p(\mathbf{n} - q_c(\mathbf{n})) \tag{5.18}$$

Similar to the described ADC approach, the distance between a query vector $\mathbf{q}$ and a database vector $\mathbf{n}$ can be calculated as follows:

$$\ddot{d}(\mathbf{q}, \mathbf{n})^2 = \sum_j d(u_j(\mathbf{q} - q_c(\mathbf{n})), q_{p_j}(u_j(\mathbf{n} - q_c(\mathbf{n}))))^2. \tag{5.19}$$

The PQ codes are learned on a set of residual vectors from the training set. Jegou et al. assume, that a single product quantizer is accureate enough when the distribution of the residuals is marginalized over all Voronoi cells. Thus, just one product quantizer is learned and used to encode the residual vectors in all cells induced by the coarse quantizer. Summarized, a novel vector $\mathbf{n}$ has to be processed in four steps for encoding:

1. assign $\mathbf{n}$ to $q_c(\mathbf{n})$
2. calculation of the residual $r(\mathbf{n})$
3. assigning $u_j(\mathbf{n})$ to $q_j(u_j(\mathbf{n}))$ for $j = 1 \dots m$
4. inserting of a new entry (identifier and code) into the inverted list of $q_c(\mathbf{n})$

A nearest neighbors search of a query $\mathbf{q}$ consists of the following steps:

1. calculation of the w closest centroids for $\mathbf{q}$ in q_c by nearest neighbor search
2. calculation of the w residuals $r(\mathbf{q})$. Note, that steps (3) and (4) are performed for each of the w cells.
3. computation of $d(u_j(r(\mathbf{q})), c_{j,i})^2 : \forall j, i$
4. calculation of the squared distance between $r(\mathbf{q})$ and indexed vectors in the inverted file by using Equation 5.19.

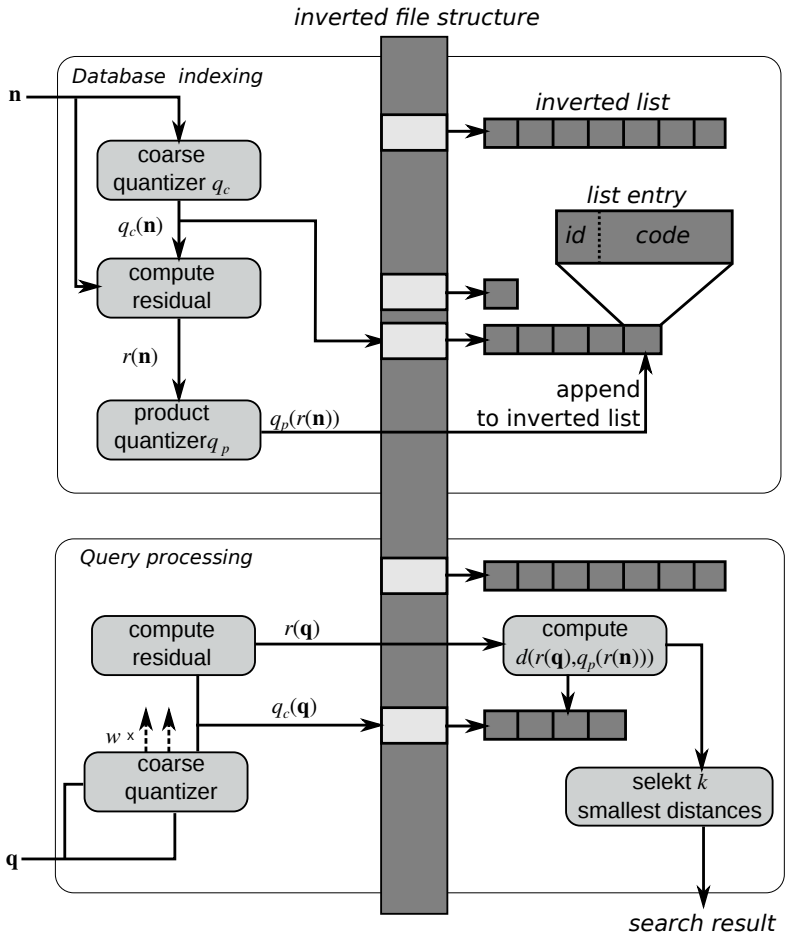


Figure 5.9: The inverted file structure adaption for the product quantization method IVFADC. In the indexing stage the vectors are quantized by a q_c and the residual vector $r(\mathbf{n})$ is encoded by a product quantizer q_p. The codes are then stored in the inverted file structure. In the search stage a predefined number w of lists is loaded and the estimated distance can be calculated. (Figure according to [41])

5. selection of the k nearest neighbors of $\mathbf{q}$ among the estimated distances.

Figure 5.9 shows an overview of the adaption of an inverted file structure for product quantization. At search time, a certain number of lists w is examined. Another important parameter is the number of centroids C' of the coarse quantizer. Lower values of C' lead to less longer lists while higher values will lead to shorter lists.

5.3.5 Comparison

This section provides a comparison of the described methods for approximate nearest neighbor search and examines comparable results reported in literature.

The FLANN methods are tree based indexing structures based on exact indexing methods using the original vectors. To allow a fair comparison between FLANN and the Product Quantizer, Jegou used its inverted file product quantizer to produce candidate lists which he resorted on basis of the original vectors. He showed a slightly better search time by equal result quality [41]. Unfortunately a direct comparison of the result quality of the PQ codes to FLANN was not given.

The three encoding techniques (SH, LSHE and PQ) can be compared in a more equitable way. To summarize briefly, all three encoding methods compress feature vectors by generating compact codes which allow faster distance computation compared to Euclidean distance computation on the original vectors. Thus, the methods facilitate increased retrieval speed but also allow to reduce the memory footprint of the vectors. Both, SH and LSHE generate binary codes. However, while SH codes sketch the overall position of the vectors, LSHE codes sketch the angles. In contrast to SH, LSHE needs the length of the vector to reconstruct an approximated distance between two vectors. LSHE and SH codes are used 'standalone' so that the codes are compared directly and no further information is needed for approximate distance calculation. In contrast to these methods, PQ needs the quantizers centroid positions to make the PQ codes comparable.

Marukatat et al. [48] evaluated the examined ANN methods on the MNIST dataset[3] which is a 60 000 images dataset for handwritten digit recognition tasks. Figure 5.10 shows the evaluation results by providing the recall rate for the methods using codes of different length. The recall rate was computed by comparison of the true rank of the 10 nearest neighbors using Euclidean distance with the methods ranking. For bit lengths higher than 128, the Product Quantizer and LSHE outperforms Spectral Hashing on this dataset for rank 10 while for shorter bit codes Spectral Hashing shows similar results. However, Jégou et al. evaluated their method on a different dataset against Spectral Hashing and report significant better recall rates for 64-bit codes throughout ranks from 1 to 1M [41]. For LSHE, the length of the vectors has to be stored along with bit codes to approximate the Euclidean distance. Marukatat et al. state, that a 32-bit double precision value is used to store the vector magnitude. However, as evaluation results are given from code lengths of 8 bit, it stands to reason that the 32 bits of the vector length are not included into the reported code length and thus qualify the reported results.

A search time of 17.2 ms is reported to compare $1M$ 8 byte PQ-codes [41]. There was no information given regarding the hardware setup though. For the calculation of the Hamming distance of $1M$ 8 byte binary codes 22.7 ms are stated [41]. For the FLANN library speed-ups by a factor of 1000 relative to a linear scan are reported preserving a recall rate of 0.95 [53].

5.4 Discussion

Algorithms to find the exact and the approximate nearest neighbors regarding their Euclidean distance in a set of vectors have been examined. The methods put their emphasis on different characteristics for nearest neighbor search. The kd-tree and R-tree try to find the exact nearest neighbors for vectors. However, the *curse of dimensionality* limits the use of these methods do

[3] http://yann.lecun.com/exdb/mnist/

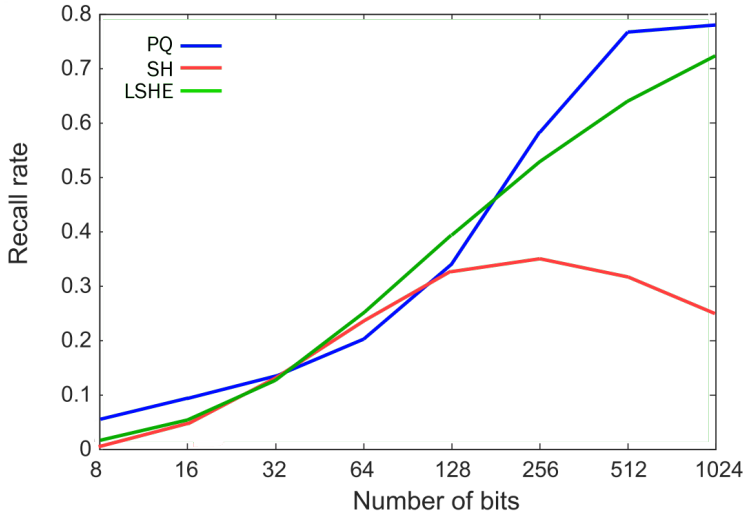


Figure 5.10: Recall rates for rank 10 from PQ, SH and LSHE for different code lengths. It stands to reason that the 32 bits needed to store the vector length in the LSHE approach are not considered for the state code lengths. (Data according to [48])

medium dimensional spaces of up to 20 dimensions. For higher dimensional spaces approximate methods can be used.

Several methods to find approximate nearest neighbors in a dataset have been examined. While FLANN tackles the problem of reducing search time by means of partitioning and traversing the feature space in an intelligent way by adapting exact indexing methods, encoding techniques concentrate on reducing the memory footprint of a dataset entry and to provide methods for comparing these codes to approximate the Euclidean distance.

Both groups of methods do not necessarily exclude each other. As the FLANN methods produces candidate lists, it can be thought of using the PQ or SH codes for resorting. The inverted file approach proposed by Jegou et al. [41] would be such an example and the usage of a hierarchical k-means clustering would be a possible alternative to the proposed coarse quantizer.

The FLANN methods do not tackle the issue of memory consumption. On the contrary, due to the overhead of the tree structures the memory footprint is increased.

Generally speaking, for all proposed methods a trade-off between search time, indexing time, search quality and memory usage has to be made. Which method to be used is application and data dependent. If all dataset vectors can be stored in main memory, the FLANN methods will probably provide good results while for very large indexes, memory aware approaches like SH, PQ or LSHE can be used [41].

In this thesis, methods that reduce the memory footprint of a record are considered only. This decision is based on the fact that due to oversegmentation (e.g. 7000 supervoxel per lung) and a moderate size of 300 volumes the number of records is in the order of millions. The exhaustive version of PQ, the non-exhaustive version IVFADC, SH and LSHE are considered as candidates for indexing supervoxel descriptors.

CHAPTER 6

Methodology, Image Descriptor Learning and Retrieval

This chapter describes the development of a texture-based image descriptor for three-dimensional medical images. The descriptor is not developed from scratch but an adaption of the *Texture Bags* approach (Chapter 3). In addition, a novel learning technique which makes use of weak image labels to reduce the semantic gap utilizing the Random Ferns approach (Chapter 4) is proposed. Finally, a retrieval and ranking method for oversegmented images is presented.

Section 6.1 gives a formal problem definition for this chapter. Section 6.2 describes an adapted version of the Texture Bag approach by utilizing Random Ferns for vocabulary building. Section 6.3 describes a weakly supervised learning technique to improve the texture based feature vectors by 'harvesting' semantic information from weak image labels. Section 6.4 describes a ranking method for the retrieval process and finally Section 6.5 discusses the developed methods.

6.1 Data, Problem Definition and Notation

The objective of this chapter is the development of an image descriptor that is able to describe texture patterns in supervoxels of medical images in order to make them comparable with respect to their visual similarity. In the following, a formal problem definition is given.

Let the (image-) database be represented by a set of I images given by $\mathcal{I} = \{\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_I\}$. For each image, there exists an oversegmentation of S_i supervoxel where $p_{i,s}$ identifies the supervoxel s in image $\mathbf{I}_i$.

We define

$$\mathcal{N} = \{p_{1,1}, \ldots, p_{1,S_1}, p_{2,1}, \ldots, p_{2,S_2}, \ldots, p_{I,1}, \ldots, p_{I,S_I}\} \tag{6.1}$$

as the set of all supervoxel identifiers in a database of I images so that $N = (\sum_{i=1}^{I} S_i)$ is the overall number of segments in the data-set. We assume, that each supervoxel represents one

of T tissue properties (e.g. 'healthy','groundglass','emphysema',...). Note, that this labeling is hidden and thus not given in the data provided. This hidden labeling can be defined by a mapping function:

$$l : \mathcal{N} \rightarrow \{1, \ldots, T\} \tag{6.2}$$

In a real world dataset, there is a set of labels $\mathcal{T}_i \subseteq \{1, \ldots, T\}$ given for each image, rather than a single label for each supervoxel. Thus, in contrast to the strong but hidden labels for supervoxels, multiple labels for images are given only:

$$\langle \mathbf{I}_i, \mathcal{T}_i \rangle_{i=1,\ldots,I} \tag{6.3}$$

That is, for each image we have a set of supervoxels and a set of labels. Let us write $\langle \mathcal{P}_i, \mathcal{T}_i \rangle$. Then

$$\forall t \in \mathcal{T}_i : \exists p_{i,s} : l(p_{i,s}) = t \tag{6.4}$$

We can also see this labeling as multiple possible labels for each supervoxel:

$$\left.\begin{array}{ll} \langle p_{1,1}, & \mathcal{T}_1 \rangle \\ & \vdots \\ \langle p_{1,S_1}, & \mathcal{T}_1 \rangle \end{array}\right] I_1$$

$$\left.\begin{array}{ll} \langle p_{2,1}, & \mathcal{T}_2 \rangle \\ & \vdots \\ \langle p_{2,S_2}, & \mathcal{T}_2 \rangle \end{array}\right] I_2$$

$$\vdots$$

Note that within each image the set $\mathcal{T}_i$ remains constant. In order to address the set of supervoxels associated with a weak label t we define $\mathcal{C}_t$ where

$$\forall p_{i,s} : (p_{i,s} \in \mathcal{C}_t) | t \in \mathcal{T}_i \tag{6.5}$$

We aim for a similarity measure d which allows to judge the visual similarity of two supervoxels. We interpret d as a distance function, so that the similarity-correlation is modeled in an inverse way. We will optimize d to come as close to the desired property of:

$$d(p_{i,s}, p_{j,u}) < d(p_{i,s}, p_{k,v}) \mid p_{i,s} = l(p_{j,u}) \wedge l(p_{i,s}) \neq l(p_{k,v}) \tag{6.6}$$

Furthermore, we also aim to find a function dv that effects image ranking similarity:

$$\forall j, k : dv(q, \mathbf{I}_j) < dv(q, \mathbf{I}_k) \mid (l(q) \in \mathcal{T}_j) \wedge (l(q) \notin \mathcal{T}_k) \tag{6.7}$$

where q is a query (e.g. a supervoxel). Less formally, we are looking for a ranking function, that allows to find images containing supervoxels with the same anomaly as a query supervoxel. Table 6.1 gives an overview of the notations defined.

Notation	Description
$\mathbf{I}$	Image
$\mathcal{I} = \{\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_I\}$	Set of I images
S_i	Number of supervoxels in image i
$N = (\sum_{i=1}^{I} S_i)$	Number of all supervoxels in the database
$p_{i,s}$	Identifier of supervoxel s in image i
$l(p)$	Hidden labeling of a supervoxel
$\mathcal{T}_i$	Weak labels of image i
$\mathcal{C}_t$	Set of all supervoxels in images with label t

Table 6.1: Notation summary

6.2 Feature Extraction with Texture Bags and Random Ferns Vocabulary

Chapter 3 examines the model of *texture bags* which is a reinterpretation of the bag-of-words or bag-of-features approach [18]. Several factors limit the suitability of this approach for our need (See Section 3.6). Therefore, we slightly modify the methodology of Burner et al. to better suit our requirements . While the overall idea has remained the same, changes in the extraction of the texture features are conducted and a random vocabulary approach for constructing the texture vocabulary is used to replace the k-means clustering for the sake of efficiency.

6.2.1 Voxel based Texture-Descriptor Extraction with LBP

The $\mathbf{d}_{LBP3d/CI}$ descriptor is examined in detail in Section 3.2. To recapture concisely, the three dimensional LBP descriptor allows to represent texture information for a $3 \times 3 \times 3$ voxel cube. In addition, $\mathbf{d}_{LBP3d/CI}$ incorporates a contrast and an intensity measure.

Initial experiments showed, that the computation of the contrast measure takes the largest share in the feature extraction process, so that we omit the calculation of the value, hoping, that the average intensity on multiple scales conveys enough information for our purpose.

In the examined method, the descriptor $\mathbf{d_{IS}}$ is scaled to the range $[0, 1]$ and γ is used to determine the impact of the measure and should be chosen according to the imaging modality. Imaging methods like CT do not have a defined maximum value as it is an encoding of material intensity, quantized into Hounsfield units [42]. A single, very high intensity voxel in the training set, used to determine the maximum value for scaling, would flaw the descriptor because the impact of the lower intensity values in the final descriptor would be reduced. Thus, we do not rescale the intensity values to $[0, 1]$ but define a more robust scaling. If $\mathcal{D}_I$ is a training set of unscaled intensity values and $\mathbf{d}_I$ is an unscaled intensity descriptor, the scaling is done as follows:

$$\mathbf{d}_{IS} = \frac{\mathbf{d}_I - min(\mathcal{D}_I)}{Q_{0.95}(\mathcal{D}_I - min(\mathcal{D}_I))}, \tag{6.8}$$

where $Q_{0.95}(.)$ denotes the 0.95 quantile of the intensities in the training set. Finally, the voxel

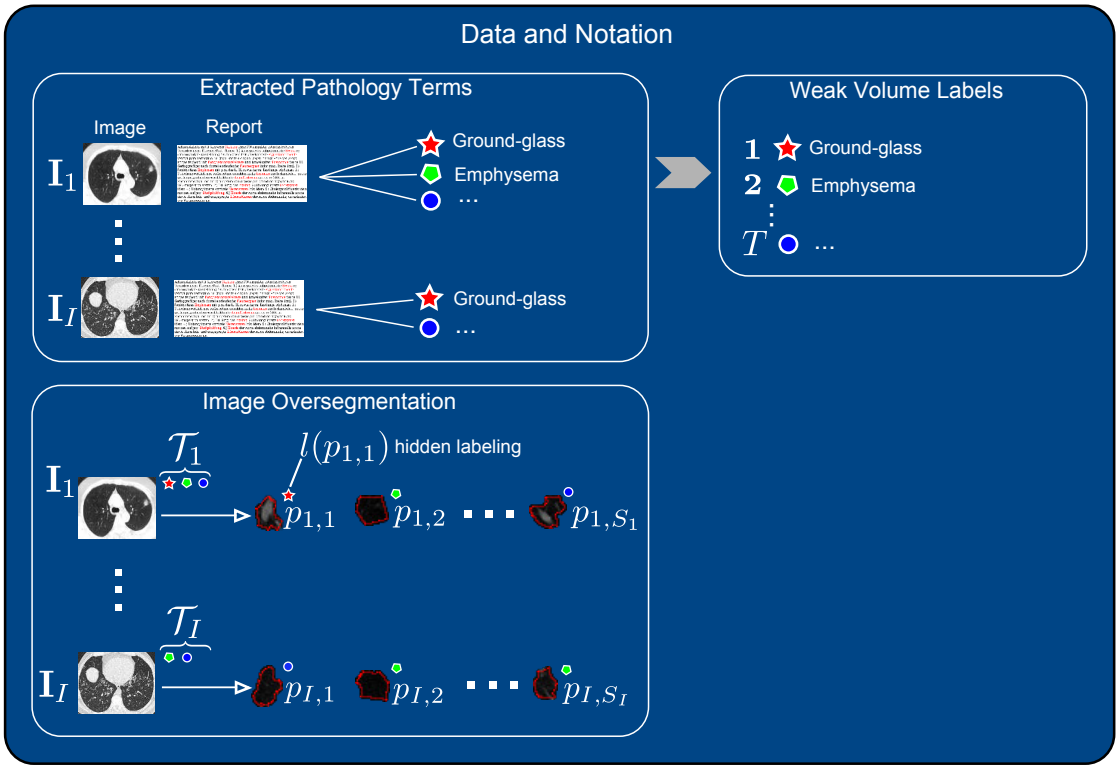


Figure 6.1: Overview of Data and Notation. Radiology terms associated with pathologies, extracted from radiological reports are used as weak labels $\{1, \ldots, T\}$ for the images. An oversegmentation of the images called supervoxels is given. The true labels $l(p_{i,s})$ describing a supervoxels tissue property is hidden.

based descriptor LBP3d/I used in this work is defined by

$$\mathbf{d}_{LBP3d/I} = [\mathbf{d}_{LBP3d}, \gamma \mathbf{d}_{IS}] \tag{6.9}$$

The final descriptor $\mathbf{d}_{LBP3d/I}$ consists of 26 dimensions for the LBP bits and 1 dimension for the intensity measure, resulting in a 27 dimensional feature vector for a voxel.

6.2.2 Vocabulary Building by Random Ferns for Texture Bags

A definition of a visual vocabulary is given in Section 3.4. Initial experiments described in Section 3.6 reveal negative runtime effects when it comes to assigning voxels to visual words by calculating the Euclidean distance. This problem is caused by the relatively large number of voxels (due to cubic growth rate) in three dimensional medical images compared to pixels in two dimensions. The main motivation behind the random vocabulary approach (Section 4.2) is the runtime improvement for vocabulary construction [52, 62]. However, the computational time needed to assign voxel descriptors to words profits from this approach as well as the nearest neighbor search is replaced by binary-test functions (See Section 8.4.2).

In this work, we construct a random vocabulary according to Definition 4.1 by means of random ferns. Figure 6.2 illustrates the process to construct such a vocabulary. The first step in constructing a vocabulary is the extraction of training examples (Steps **(1)** and **(2)** in Figure 6.2). For feature space partitioning, we use a splitting function, operating on random sub-spaces and threshold splitting according to Equation 4.12 (Step **(3)**). The resulting partitioning can be used to quantize the training examples (Step **(4)**).

The dimensionality of a BVW image descriptor corresponds directly to the vocabulary size (as long as no feature selection or dimensionality reduction is applied). Therefore, the size of a vocabulary is an important property to consider for further processing (learning, indexing, clustering...) of the resulting descriptors.

The size of a random vocabulary increases exponentially over the fern depth but parts of the words cover regions in feature space where no or very few (outliers with no discriminative power) examples exist. Like proposed by Mu et al. [52], the vocabulary is trimmed back to a predefined value W by introducing a mapping-table $S : \{0,1\}^L \rightarrow \{1,..,W\}, W \leqslant 2^L$ that selects the W words with the highest frequency in the training set (Step **(5)**).

Initial experiments (See Section 8.4.2) show, that this way of constructing a random vocabulary can have negative properties. Due to the randomness of the approach, the chance, that distribution of features to words becomes unevenly (in plain words: few words with high frequency and many words with low frequency) is given. This may be desirable to a certain degree as low frequency words are more discriminative [83]. However, the power to describe more general variations is reduced [83]. In the following, an approach to diminish this effect is given.

Candidate Vocabulary Selection: To reduce the effect of unevenly distributed word frequencies, we construct R independent random vocabularies and choose the one with most homogeneous word frequencies. In order to do so, we calculate the variance of the histograms of the training words in the reduced vocabulary (after application of the mapping table S) and choose the one with the lowest variance (Step **6**). The variance of the histogram as quality measure is rectified by the idea, that a vocabulary with variance 0 would uniformly quantize the training features.

Figure 6.2 illustrates this vocabulary training process.

Following the multi-scale approach examined in Section 3.3, independent vocabularies on E differently scaled images are trained. The resulting vocabularies are implicit and no centers or word candidates are given so that a vocabulary V_e is represented by a vocabulary fern $\mathcal{F}_e^V$, and mapping $\mathcal{S}_e^V$:

$$V_e = \langle \mathcal{F}_e^V, \mathcal{S}_e^V \rangle \; \forall e \in \{1 \dots E\} \tag{6.10}$$

6.2.3 Histograms Binning of Texture Words on multiple Scales

Let $\mathcal{D}_e$ bet the set of LBP3d/I descriptors extracted from a supervoxel on scale e. After vocabulary construction, the features of a supervoxel are assigned to visual words by applying the trained random fern and the mapping table according to Equation 3.3:

$$\mathbf{d}_{LBP3d/I} \mapsto w \in \mathcal{W}, \; \forall \mathbf{d}_{LBP3d/I} \in \mathcal{D}_e \tag{6.11}$$

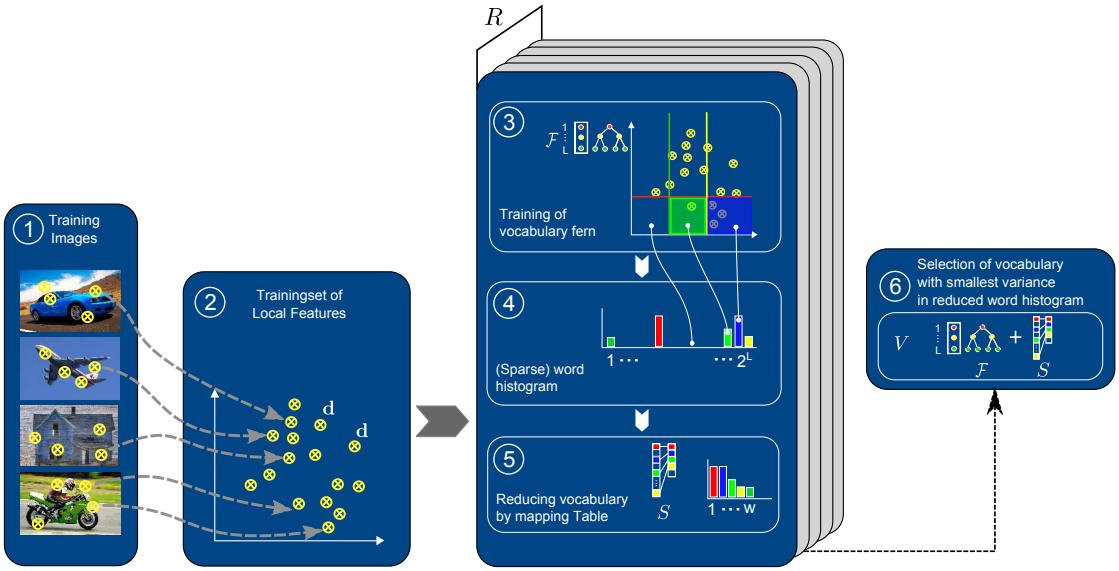


Figure 6.2: Overview of the vocabulary generation process by random ferns. At steps (1) and (2) a training set of local features is extracted from a training set of images. (3), a random partitioning of the feature space is performed by random ferns. (4), the features are summed into a histogram by counting the occurrences in the partitions. (5), the histogram is trimmed back to a fixed size. Steps (3) (4) and (5) are repeated R times. (6), the fern and mapping table, yielding the reduced histogram with lowest variance are selected to be the final vocabulary.

Following the idea in Chapter 4, the concatenated histogram

$$h(\mathbf{p}) = [\alpha_1 h_1(p) \dots \alpha_E h_E(p)] \tag{6.12}$$

of the words in a supervoxel is used to represent the supervoxel. In addition to the texture bags model, the weighting factors $\alpha_1, \dots, \alpha_s$ to control the impact of the different scales are implemented. Subsequently, the histograms are normalized to sum 1 to allow the comparison of two supervoxels without considering their size. We define $\mathbf{f}^{TB}(p)$ as an image descriptor of supervoxel p by

$$\mathbf{f}^{TB}(p) = h(p). \tag{6.13}$$

6.2.4 Distance Measure and Dimensionality Reduction

In Chapter 5, several methods to search in a collection of vectors are examined. Promising (See Section 5.4) ANN methods like PQ or SH approximate L_p metrics (e.g. Euclidean distance) between vectors. Therefore, we decided to use Euclidean distance to judge the similarity of supervoxels. We interpret any descriptor $\mathbf{f}^{TB}$ as a point in the Euclidean space $\mathbb{R}^d, d = S \times W$ so that $\mathcal{N}^{TB} \subset \mathbb{R}^d$. In order to address the set of all feature vectors in the dataset we define:

$$\mathcal{N}^{TB} = \{\mathbf{f}_1^{TB}, \dots, \mathbf{f}_2^{TB}, \dots, \mathbf{f}_N^{TB}\} \tag{6.14}$$

where $\mathbf{f}_{i,s}^{TB} = \mathbf{f}^{TB}(p_{i,s}) = h(p_{i,s})$. Even if designed for very high dimensions, the performance of ANN methods degrades with increasing dimensionality [53]. We compare two different dimensionality reduction techniques to diminish the curse of dimensionality (Section 5.2.3), reduce the memory footprint of a record and allow faster distance computation between two entries. The methods Principal Component Analysis (PCA) and Random Projections (RP) for dimensionality reduction are described in the following.

Dimensionality reduction via PCA

PCA is a dimensionality reduction technique based on projection onto a subspace which is spanned by some orthonormal basis [69]. This basis is build by the most important eigenvectors of the covariance matrix of the input vectors. In order to allow the calculation of the principal components one has to center the set of vectors used to 0 by subtracting the mean of each dimension. Subsequent, the vectors $\mathcal{N}^{TB}$ are arranged in a matrix $\mathbf{N}_{d\times N}$ and the following steps are performed:

- Calculation of the covariance matrix

$$\mathbf{C} = \frac{1}{n-1}\mathbf{N}\mathbf{N}^{\top} \tag{6.15}$$

- extraction of a set of d orthonormal eigenvectors $\langle x_1 ... x_d \rangle$ and eigenvalues $[\lambda_1 ... \lambda_d]$ from $\mathbf{C}$
- sorting the eigenvectors according to the decreasing size of the eigenvalues
- selection of the first d^* eigenvectors $\mathbf{V} = \langle x_1 ... x_{d^*} \rangle$ which represent a linear sub space, where d^* is the target dimensionality

The eigenvectors given, any vector $\mathbf{f}^{TB} \in \mathbb{R}^d$ can be projected into the embedding space by

$$\mathbf{f}^{PCA} = \mathbf{V}^{\top} \cdot \mathbf{f}^{TB} \tag{6.16}$$

In order to retrieve the principal components, just a subset of the vectors may be used to reduce computation time.

Dimensionality Reduction via Random Projections

RP allows to reduce the dimensionality by projecting the data of dimensionality d to a lower, randomly generated subspace of dimension d^* [13]. In order to do so, a $d \times d^*$ matrix $\mathbf{R}$ of random values, sampled from the standard normal distribution, is generated and the columns are normalized to be of unit length. Subsequent, a vector $\mathbf{f}^{TB} \in \mathbb{R}^d$ can be projected onto the subspace by

$$\mathbf{f}^{RP} = \mathbf{R}^{\top}\mathbf{f}^{TB} \tag{6.17}$$

In contrast to PCA, vectors of the projection matrix are generated independently of the data. Its approximation property is based on the John-Lindenstrauss lemma, which states, that an

orthogonal projection to a lower dimensional space preserves the distance between two points with respect to a certain distortion factor [77]. Thus the random matrix $\mathbf{R}$ would ideally hold orthogonal vectors. In a high-dimensional space, the exact orthogonality might be ignored as there exits a sufficient large number of vectors in almost orthogonal direction [34]. In our case, we did not orthogonalize the vectors in matrix $\mathbf{R}$.

6.3 Weakly Supervised Descriptor Learning

In the previous section, the feature vectors $\mathbf{f}^{TB}$, $\mathbf{f}^{RP}$ and $\mathbf{f}^{PCA}$ are defined. Each of these descriptors is designed to represent texture properties of supervoxels. As these descriptors are based on the texture bag histogram h, application specific descriptors, relying on intensity information are constructed. We will define a distance that takes semantic information into account by linking the descriptors to the weak labels (Equation 6.3). This enables us to decrease the distance between supervoxel descriptors with similar labels.

6.3.1 Outline and Idea

A visual descriptor (e.g. *Texture Bags* in Section 6.2) is comprised of a number of values and can be interpreted as vector in the Euclidean space. Each dimension represents a visual feature (e.g visual word or texture word). Not all features may be needed to describe the content of the image in a certain context. Each anomaly label t can be seen as a class label, represented by a subset of texture patterns. We assume that the distribution of the descriptors of members of a class are concentrated in certain areas of the feature space. A challenging property of our data is the number of weak labels extracted from radiological reports, that may be relatively high (in a set of 327 lungs, 175 terms associated with anomalies are extracted). We expect the data to be noisy and their occurrences may vary strongly (See Figure 8.4).

To link the weak label information to the local appearance features, we develop a novel supervoxel descriptor, comprised of indicators for a set of terms. Each indicator reflects the affinity of the supervoxel to a certain term. We assume, that such a vector, learned on a sufficiently large number of terms, reflects the visual properties that are important to distinguish between anomaly classes. The distance between these vectors of indicators is then used to judge visual similarity.

In the following, an approach, based on the ideas of Random Ferns ensemble for classification (Section 4.1) and Probability Estimation Trees (PET) [16] is presented. PETs are classification trees with a class probability distribution at each leaf instead of a single class label [16]. A ferns ensemble learns indicators, reflecting visual similarity with respect to class label distribution in the feature space.

6.3.2 Subspace Partitioning by Random Ferns

This subsection describes the approach used to learn a model of the class label distribution in the feature space. The texture bags descriptors $\mathbf{f_i}^{TB}$ are used as input. Figure 6.3 gives an overview of the partitioning process described. The Random Ferns approach for classification and vocabulary learning (Section 4) is used for space partitioning. We generate an ensemble $\mathcal{E}$

of E Random Ferns of depth L to partition the feature space.

$$\mathcal{E} = \langle \mathcal{F}_1, \ldots, \mathcal{F}_E \rangle \tag{6.18}$$

The splitting function $\mathcal{L}$ defined in Equation 4.12 is used. Every node operates on a subspace of the feature space and splits the complete space. Each fern generates 2^L partitions

$$\mathcal{F}(\mathbf{f}_i^{TB}) = y, y \in \{1, \ldots, 2^L\} \tag{6.19}$$

and the ensemble generates E independent partitionings so that

$$\mathcal{E} : \mathbb{R}^d \rightarrow \langle y_1, \ldots, y_E \rangle, y \in \{1, \ldots, 2^L\} \tag{6.20}$$

$$\mathcal{E}(\mathbf{f}_i^{TB}) : \mathbf{f}_i^{TB} \mapsto \langle y_1, \ldots, y_E \rangle, y \in \{1, \ldots, 2^L\} \tag{6.21}$$

Further, we define $\mathcal{Y}_y^e$ as the set of supervoxels lying in a certain partition y of fern $\mathcal{F}_e$ so that

$$\forall i \in \mathcal{N} : i \in \mathcal{Y}_y^e \mid \mathcal{F}_e(\mathbf{f}_i^{TB}) = y \tag{6.22}$$

Lets consider following conditions after the ensemble based partitioning of the dataset:

- Each set of supervoxels $\mathcal{Y}_y^e$ of a feature space partition represents a neighborhood, thus holds a group of supervoxels which share visual properties.
- Two partitions $\mathcal{Y}_k^i$ and $\mathcal{Y}_b^j$, $i \neq j$, induced by two different ferns, intersect as they were generated on different subspaces and are therefore likely to be orthogonal to each other.

For any $\mathcal{Y}_y^e$, we analyze the frequencies of the weak labels by its relative class frequency. We define $rf(\mathcal{Y}_y^e, t)$ as the relative class frequency of t in the leaf set $\mathcal{Y}_y^e$:

$$rf(\mathcal{Y}_y^e, t) = \frac{l(\mathcal{Y}_y^e, t)}{\sum_{j=1}^{T} l(\mathcal{Y}_y^e, j)} \tag{6.23}$$

where $l(\mathcal{Y}_y^e, t)$ gives the number of examples in the leaf node $\mathcal{Y}_y^e$ which belong to class t:

$$l(\mathcal{Y}_y^e, t) = \sum_{i=1}^{N} \mathbf{1}_{\mathcal{Y}_y^e}(i) \mathbf{1}_{\mathcal{C}_t}(i) \tag{6.24}$$

The relative class frequency makes leaf nodes or partitions comparable with respect to the number of examples of a certain class they hold. Some leaf nodes may hold very few or even just a single example. A leaf with one entry would lead to the highest possible value for the examined class while a leaf with 100 examples of this class and just 1 example of another class would be ranked less likely. We use the Laplace Smoothing to solve this problem [16, 64]:

$$lp(\mathcal{Y}_y^e, t) = \frac{1 + l(\mathcal{Y}_y^e, t)}{T + \sum_{j=1}^{T} l(\mathcal{Y}_y^e, t_j)} \tag{6.25}$$

The use of lp smooths the frequencies from small samples by applying an equal prior to all classes.

Classification ferns or PET combine single tree/ferns predictions by giving each leaf, where an example falls in, the same vote (e.g. Equation 4.6 or probability estimates in [16]). Considering a weak training set with very few true representatives for a class in a class-bag, we use the ferns ensemble as a mean to approximate distances in the feature space.

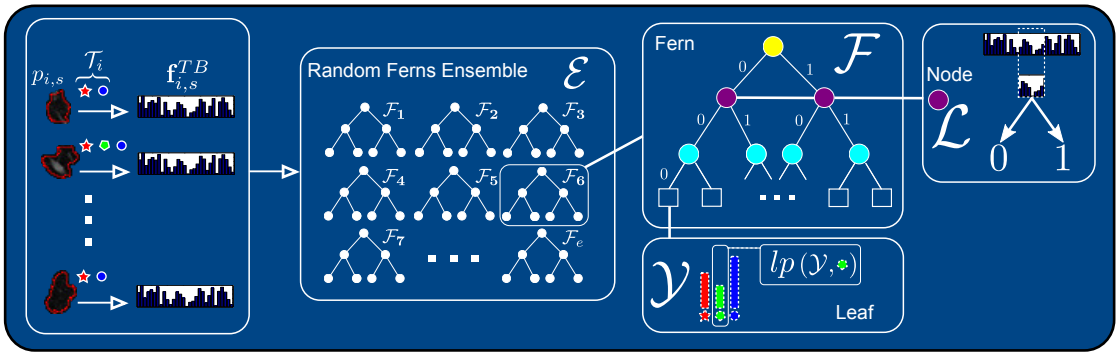


Figure 6.3: Schematic overview of random feature space partitioning and term frequency analysis. The input data is comprised of feature vectors representing supervoxels and weak labels belonging to the image where the supervoxel originates from. The feature space is randomly partitioned by an ensemble of RF $\mathcal{E}$. Each fern is composed of a set of splitting functions operating on a subspace. Fore each leaf (partition) the relative frequency can be calculated for each term.

6.3.3 Semantic Profile

Instead of using the Euclidean distance $d_{Euc}(\mathbf{f}_j^{TB}, \mathbf{f}_k^{TB})$, we can use the ferns response to define a similarity measure $d_{\mathcal{E}}(j,k)$:

$$d_{\mathcal{E}}(j,k) = \sum_{e=1}^{E} \sum_{y=1}^{2^L} \mathbf{1}_{\mathcal{Y}_y^e}(i) \mathbf{1}_{\mathcal{Y}_y^e}(j). \tag{6.26}$$

$d_{\mathcal{E}}(j,k)$ counts the ferns that put $\mathbf{f}_j^{TB}$ and $\mathbf{f}_k^{TB}$ into the same leaf node. As a leaf node represents a neighborhood, the value of $d_{\mathcal{E}}$ intuitively approximates the inverse distance in the feature space and thus the similarity of the two supervoxels.

In the next step, we consider leaf nodes with "high" relative class frequency for a certain class t only so that the resulting similarity measure overrates dimensions which are associated with this class.

Let us define

$$\mathcal{K}^t = \{\langle e_1, y_1 \rangle, \ldots, \langle e_K, y_K \rangle\} \tag{6.27}$$

as the set of the K leaf nodes in the ensemble with the K highest relative class frequencies for class t. Counting the occurrences in these nodes only, we define a new similarity measure between two voxels:

$$ds_{\mathcal{E}}(j,k,t) = \sum_{e=1}^{E} \sum_{y=1}^{2^L} \mathbf{1}_{\mathcal{Y}_y^e}(j) \mathbf{1}_{\mathcal{Y}_y^e}(k) \mathbf{1}_{\mathcal{K}^t}(\langle e, y \rangle). \tag{6.28}$$

Let consider a hypothetical supervoxel that is represented in all leaf nodes in $\mathcal{K}^t$. This supervoxel could be seen as a prototype representative of class t. We see the similarity of a supervoxel to this prototype according to $ds_{\mathcal{E}}$ (Equation 6.28) as a measure of its affinity to the class.

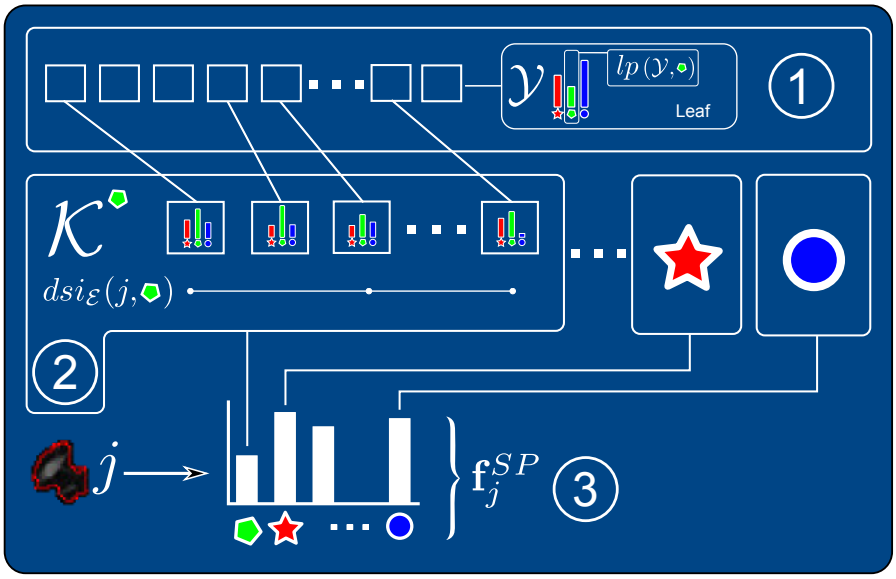


Figure 6.4: Overview on the calculation of the Semantic Profile descriptor. **(1)**: Feature space partitioning and calculation of the smoothed relative class frequency according to Subsection 6.3.2. **(2)**: Counting the occurrences of a supervoxel j in the K partitions with the K highest frequencies for a certain class. Step 2 is performed for all classes. **(3)**: The Semantic Profile descriptor is the concatenation of the normalized $dsi_{\mathcal{E}}(.,.)$ values for the classes used.

Based on Equation 6.28, we construct an affinity indicator for a database supervoxel j reflecting its affinity to any t:

$$dsi_{\mathcal{E}}(j,t) = \sum_{e=1}^{E} \sum_{y=1}^{2^L} \mathbf{1}_{\mathcal{Y}_y^e}(j) \mathbf{1}_{\mathcal{K}^t}(\langle e, y \rangle)). \tag{6.29}$$

$dsi_{\mathcal{E}}(j,t)$ sums the occurrences of a supervoxel in $\mathcal{K}^t$ so that $dsi_{\mathcal{E}}(j,t) \in 0,\ldots,E$.

For a supervoxel j, $dsi_{\mathcal{E}}(j,t)$ is calculated for $t = 1,\ldots,T$ resulting in a new vector. The maximum value for one class reached in a dataset is E. In practice, this value will be smaller and vary between different classes. Thus, we scale each dimension to $[0,1]$ resulting in the new feature vector that we call *Semantic Profile* (SP):

$$\mathbf{f}_j^{SP} \in [0,1]^T = [dsi_{\mathcal{E}}(j,1),\ldots,dsi_{\mathcal{E}}(j,T)] \times [\max_i(dsi_{\mathcal{E}}(i,1)),\ldots,\max_i(dsi_{\mathcal{E}}(i,T))]^{-1} \tag{6.30}$$

Figure 6.4 gives an overview of the SP generation.

We expect, that this descriptor provides a higher performance in retrieving anomalies compared to feature vectors relying on visual information only and in addition reduce the feature space dimensionality to T.

6.4 Retrieval and Ranking on the Image Level

The subervoxel segmentation of the images and descriptor extraction allows to establish a nearest neighbor search for supervoxels based on ranking of the (e.g. Euclidean) distance between a query and a database supervoxel descriptor:

$$d(\mathbf{q}, \mathbf{f}_i) = d_{Euc}(\mathbf{q}, \mathbf{f}_i). \tag{6.31}$$

$\mathbf{f}$ is a place-holder for any of the vectors ($\mathbf{f}^{PCA}, \mathbf{f}^{RP}$ and $\mathbf{f}^{SP}$) presented in this chapter. We do not only retrieve and order supervoxels in the dataset but also need a way to order images. It is not in the scope of this work to compare different ranking algorithms or to optimize the presented approach but rather to show that there is a way to rank images based on supervoxel descriptor distances.

In this work, we interpret a query as a set of supervoxels represented by their feature vectors. This section describes a way to order images based on a list of retrieved descriptor distances.

The following two conditions are considered for the ranking algorithm:

- The result of a single supervoxel query search is not complete. We obtain just the k-NN for a query rather than a sorted list of all supervoxels in the database. Thus, the count of supervoxel memberships to images may vary in the supervoxel result lists (e.g. One image is represented by just 100 supervoxels while another image by 2000).
- Not the whole image but rather a part of a result image may be visually similar to the queries (e.g. a query represents an anomaly that just occurs in subregions of images).

The queries are given by a set of descriptors $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_Q\}$. After k-NN search is performed for each of the queries, a joint list of the nearest neighbors of the queries $\mathcal{M} \subset \mathcal{N}$ is given.

Based on this joint list ranking of the images is performed in two steps:

1. For each supervoxel in $\mathcal{M}$, the average distance between the queries and the supervoxel is calculated. For any $p_{i,x} \in \mathcal{M}$, the average distance is calculated by:

$$\bar{ds}(\mathcal{Q}, \mathbf{f}_{p_{i,x}}) = \frac{\sum_{q=1}^{Q} d_{Euc}(\mathbf{q}_q, \mathbf{f}_{p_{i,x}})}{Q} \tag{6.32}$$

   If a supervoxel is not present in the result list of a query, the maximum distance among the retrieved supervoxels is used.

2. Let $\mathcal{R}_i \subset \mathcal{M}$ be the set of supervoxels retrieved that belong to an image i so that $\forall \mathbf{f}_{j,s} \in \mathcal{R}_i | i = j$. The value used for ranking the image is calculated by:

$$\bar{d_{\mathcal{I}}}(\mathcal{Q}, i) = \frac{\sum\limits_{x \in \mathcal{R}_i} \bar{ds}(\mathcal{Q}, x)}{|\mathcal{R}_i|} \cdot \frac{c}{\min(|\mathcal{R}_i|, c)} \tag{6.33}$$

   c is a parameter to discriminate images that are represented by a relative small number of supervoxels in the result list. This step is rectified, as it may not be desirable to judge the presents of an anomaly in an image on the basis of a few supervoxels.

On the basis of the described metric, a ranking of images can be performed for a set of query descriptors.

6.5 Discussion

This chapter described the development of texture based image descriptors. *LBP3d/i* is used to encode local gray-value gradients in 3x3x3 voxel cubes. The method of RF is us to create an implicit visual vocabulary. The implemented random vocabulary differs from the method of Mu et al. [52] in the points that (1) the splitting function proposed by Pauly et al. is used and (2) that we create multiple candidate vocabularies and used the one with the lowest variance of word frequencies. Two alternative techniques (PCA and RP) are applied to reduce the dimensionality of the texture bag descriptor. A method to link visual information (texture bags) and textural information (image labels) is presented. The resulting SP descriptor is a vector of values that reflects a supervoxels affinity to the image labels.

A way to rank images on the basis of multiple supervoxel queries is described. However, more complex ranking methods are possible. In the following, a list of possible features that a image ranking metric could offer is given:

- The physical distance between the supervoxels could be considered to find connected regions of similar supervoxels in the target images.
- The size of the query (# query supervoxels) could be used to set threshold c in Equation 6.33 (e.g. the threshold increases with the number of query supervoxels).
- The metric could consider heterogeneous queries (e.g. representatives of visually different anomalies) to retrieve compositions of anomalies in images.
- Metric parameters could be set by the user (e.g. expanse of target structure)

CHAPTER 7

Methodology, Indexing

In Chapter 6, the calculation of the Euclidean distance between image descriptors is used to judge the similarity of supervoxels. To retrieve a set of visually similar supervoxels from a dataset, nearest neighbors of a query vector have to be found in the database-set of image descriptors. This chapter describes the application of multiple indexing methods in order to facilitate faster search compared the naive exhaustive computation of the distance to all entries in the dataset.

Several indexing methods are examined in Chapter 5. The common idea of any indexing method is to invest computation time in advance and/or additional memory to facilitate improved search time. Two groups of methods (exact and approximate) nearest neighbor search methods are described. In this work, ANN (Section 5.3) methods encoding vectors to compact representatives are applied only.

The decision to use encoding methods in this work rather than exact methods or FLANN is based on following considerations:

1. The dimensionality of the texture descriptors used is higher than 20. Thus, exact methods are not applicable (See Section 7.1).

2. Encoding methods allow to reduce the memory footprint of a supervoxel descriptor in the database.

3. Distortions in the nearest neighbor result list may be delimited by subsequent image ranking. The effect of distortions is analyzed in Section 9.2.

In addition to the encoding techniques described in Section 5.3 the idea to use RF for binary code generation is investigated.

In the following, the application of the methods used is described. Section 7.1 describes SH for indexing, Sections 7.3 and 7.3 the exhaustive and non-exhaustive versions of PQ. Section 7.2 uses LSHE and Section 7.5 describes a way to use RF for binary code generation.

7.1 Indexing by means of Spectral Hashing

The method of SH [81] is described in detail in Section 5.3.2. Given an image descriptor $\mathbf{f} \in \mathbb{R}^d$, SH generates a binary code of a certain length d_b:

$$sh : \mathbb{R}^d \rightarrow \{0,1\}^{d_b} \tag{7.1}$$

so that $sh(\mathbf{f}_i) : \mathbf{f}_i \mapsto \mathbf{sh}_i$. In order to generate the embedding, Equations 5.7 and 5.8 are applied on the set of vectors. SH is designed in a way, that the Hamming distance $h(\mathbf{sh}_i, \mathbf{sh}_j)$ (Equation 5.9) reflects the Euclidean distance $d_{Euc}(\mathbf{f}_i, \mathbf{f}_j)$. We define

$$d_{SH}(\mathbf{q}, \mathbf{f}_i) = d_{Ham}(sh(\mathbf{q}), \mathbf{sh}_i) \tag{7.2}$$

as an alternative distance computation to Equation 6.31.

In practice, the binary hash codes are stored as a set of 8-bit unsigned integer values [81]. To calculate the binary Hamming distance the Exclusive Or Product (XOR) of the integers is built. The Hamming distances for all combinations of 8-bit values is precalculated and stored in a 256x256 lookup table so that the binary Hamming distance calculation for a set of bytes is performed by lookup and addition.

7.2 Indexing by means of Local Sensitive Hashing for Euclidean

The method of LSHE [48] is described in Section 5.3.3. LSHE generates binary codes from image descriptors by applying a set of d_b binary tests of the form presented in Equation 5.10:

$$lsh : \mathbb{R}^d \rightarrow \{0,1\}^{d_b}. \tag{7.3}$$

Like SH, LSH generates a mapping $lsh(\mathbf{f}_i) : \mathbf{f}_i \mapsto \mathbf{lsh}_i$. In order to approximate the Euclidean distance between two vectors, the length of the vector is stored as well, so that

$$lshe_i = \langle \mathbf{lsh}_i, \|\mathbf{f}_i\| \rangle \tag{7.4}$$

The distance between two LSHE sets is calculated according to Equation 5.14:

$$d_{LSH}(\mathbf{q}, \mathbf{f}_i) = \|\mathbf{q}\|^2 + \|\mathbf{f}_i\|^2 - 2\cos(\frac{\pi}{m} d_{Ham}(lsh(\mathbf{q}), \mathbf{lsh}_i)) \tag{7.5}$$

We use a single precision floating point value to store the vector length so that the overall bit length of a $lshe_i$ descriptor is $d_b + 16$. To calculate the Hamming distances the approach described in Section 7.1 is used.

7.3 Indexing by means of Product Quantization

PQ [41] is described in Section 5.3.4. To quantize the m subspaces (subquantizer), k-means clustering (Section 3.4.1) with $C*$ cluster center is used so that m centroid indices are calculated

for a database vector $\mathbf{f}_i$. The application of PQ is a mapping function, mapping a vector to a set of integers:

$$PQ : \mathbb{R}^d \rightarrow \{1, \ldots, C*\}^m \tag{7.6}$$

so that $\mathbf{f}_i \mapsto \mathbf{pq}_i$. Besides the $\mathbf{pq}$ integer vectors, the $m.C*$ cluster centers are stored. To compute the approximate distance between a vector and the database entries the ADC distance computation $\tilde{d}(.,.)$ (Equation 5.16) is used. We define

$$d_{PQ}(\mathbf{q}, \mathbf{f}_i) = \sqrt{\sum_{s=1}^{m} d_{Euc}(u_s(\mathbf{q}), q_{p_s}(u_s(\mathbf{f_i})))^2} \tag{7.7}$$

where $u_s(.)$ is the subspace and $q_{p,s}(u_s(\mathbf{f_i})) = \mathbf{c}_{s,r}$ is the centroid associated with the integer number stored in $\mathbf{pq}_i^{[s]}$. The distances $d_{Euc}(u_s(\mathbf{q}), \mathbf{c}_{r,s}))^2$ for all $r = 1, ..., C*$ and $s = 1, ..., m$ are calculated in advance.

7.4 Indexing by means of Inverted File and Product Quantization

Jégou et al. report better distance approximation for their non-exhaustive version of PQ compared to the exhaustive version [41]. In order to validate and quantify this effect for our data, the InVerted File Product Quantization (IVFPQ) is applied. To perform the preliminary coarse quantization, k-means clustering is used again. According to Equation 5.17 the residual vectors $r(\mathbf{f}_i) = \mathbf{f}_i - q_c(\mathbf{f}_i)$ to the centroids of the coarse quantizer are encoded by PQ. According to Equation 5.18 a descriptor $\mathbf{f}$ is approximated by:

$$\ddot{\mathbf{f}} \triangleq q_c(\mathbf{f}) + q_p(r(\mathbf{f})) \tag{7.8}$$

The version of the ADC distance computation described in Equation 7.1 is used to determine the distance to a novel vector:

$$d_{IVFPQ}(\mathbf{q}, \mathbf{f}_i) = \sqrt{\sum_{s=1}^{m} d_{Euc}(u_s(\mathbf{q} - q_c(\mathbf{f}_i)), q_{p_s}(u_s(\mathbf{f}_i - q_c(\mathbf{f}_i))))^2} \tag{7.9}$$

Again, $q_{p_s}(u_s(\mathbf{f}_i - q_c(\mathbf{f}_i)))$ is a centroid $\mathbf{c}_{r,s}$ to which the distances are calculated preliminary.

7.5 Indexing by means of Random Ferns

SH is an example of a method that uses binary tests in an intelligent way, considering structure in the distribution of the data to be indexed. In this section, the use of RF as an alternative way to construct binary codes for ANN-search in Hamming space is investigated. Binary tests in the form of random hyperplanes by means of RF are applied to generate binary codes to represent vectors.

To generate codes of a certain length d_b, a fern of depth d_b is applied:

$$\mathcal{F}(\mathbf{f}_i) : \mathbf{f}_i \mapsto \mathbf{rf}_i \tag{7.10}$$

where $\mathbf{rf} \in \{0,1\}^{d_b}$.

The nearest neighbor search is performed identical to SH codes by calculating the Hamming distance between the binary codes of the query and the database vector. Thus, the binary tests have to be applied on the query vector preliminary to the search:

$$d_{RF}(\mathbf{q}, \mathbf{f}_i) = d_{Ham}(\mathcal{F}(\mathbf{q}), \mathbf{rf}_i) \tag{7.11}$$

The number of binary tests influences the quality of the approximation as every split adds information and due to the random nature of the approach, the stability of the approximation is influenced by code length as well. Considering search speed and memory consumption, it is desirable to produce codes of the shortest length which still meet the target applications quality requirements. Weiss et al. [81] states, that a binary code for approximation is efficient, if each bit has a 50% chance of being one or zero but also that the bits are independent of each other. In contrast to SH, splits by RF are random and not performed with respect to such a criteria. Thus, we investigate a way to generate binary codes by random splits but shorten the sequence by using some quality criteria to reduce the code length.

Splitsequence Decomposition: We suggest a way to improve the efficiency of random codes by selecting "good" binary sequences. In the vocabulary training process described in Section 6.2.2 a quality criteria for a vocabulary was introduced to select the partitioning that most uniformly partitions the training examples. This is done by using the variance of the word frequencies in the partitions. While this is possible for a relatively low number of splits (≤ 16) like it is used for random vocabularies, the exponentially growing number of partitions over the number of splits makes this approach infeasible for longer binary codes (e.g. 32-bit codes produce $> 4 * 10^9$ partitions). To relax this quality criteria in order to make it applicable for longer codes only short (e.g. length 8) sequences of binary splits are analyzed and the concatenation of sequences with lowest partition-size-variance is used to compose the final split sequence. In the following, this approach is described more formally.

Let s be the number of splits for which a resulting space partitioning can be analyzed (e.g. 8). Analogue to the presented random vocabulary approach (Section 6.2.2), $R \gg \frac{d_b}{s}$ split sequences (subferns) of length s are produced. Each of this sequences constitutes a fern $\mathcal{F}^{[s]}$ itself. For each of this ferns a histogram of entries over the resulting leaf indizes is created. Like in the random vocabulary approach, the variance of those histograms is used to judge how much approximation power this partitioning can contribute. The final fern for code generation is generated by concatenation of the $\frac{d_b}{s}$ subferns that show lowest histogram variance.

7.6 Discussion

In this chapter, the application of several encoding techniques to index image descriptors is described. The methods of SH, LSHE, PQ and IVFPQ are performed according to the steps described in the State of 5. The idea to use RF to generate binary codes which facilitate nearest neighbor search in Hamming space is proposed. The suitability of the described methods to be used for retrieval is determined by experiments described in Section 9.2.

CHAPTER 8

Evaluation and Setup of Experiments

This chapter describes the data used for evaluation (Section 8.1), query definition and extraction (Section 8.2), evaluation measures (Section 8.3) and the experimental set-up and parameter configuration (Section 8.4).

The goal of the evaluation is to show the ability of the implemented descriptor and indexing methods to rank the defined information need in a retrieval system. Given a query, the information need is a ranked list of images which contains visually similar regions. To measure information retrieval effectiveness, the following is required [47]:

- A collection of records.
- A set of queries as expression of the information needs.
- A set of relevance judgments which can be a binary assessment of either relevant or non-relevant for each query-record pair.

8.1 Test-Collection Dataset and Preprocessing

All experiments are conducted on a dataset of 318 thorax CT images with attached radiological reports. All images are generated in the daily routine of a hospital. Thus, properties like device manufacturer, image resolution and reconstruction kernel may differ and are not known in our case. To allow consistent processing, all volumes are transformed to an isotropic voxel resolution of 0.7mm in advance. For each volume, an oversementation is created by means of a 3 dimensional adaption of the mono-SLIC superpixel algorithm [36]. The size of supervoxels created by the algorithm is set to $1cm^3$.

The experiments are conducted on images showing lungs. Thus, feature extraction and search are performed on supervoxels which are at least partially within the lung. To enable that, segmentation of the lungs is performed by a combination of semi-automatic threshold algorithm, application of simple morphological functions and manual validation. Lung-segmentation is not in the scope of this work. The algorithm and the implementation used was developed at CIR-Lab

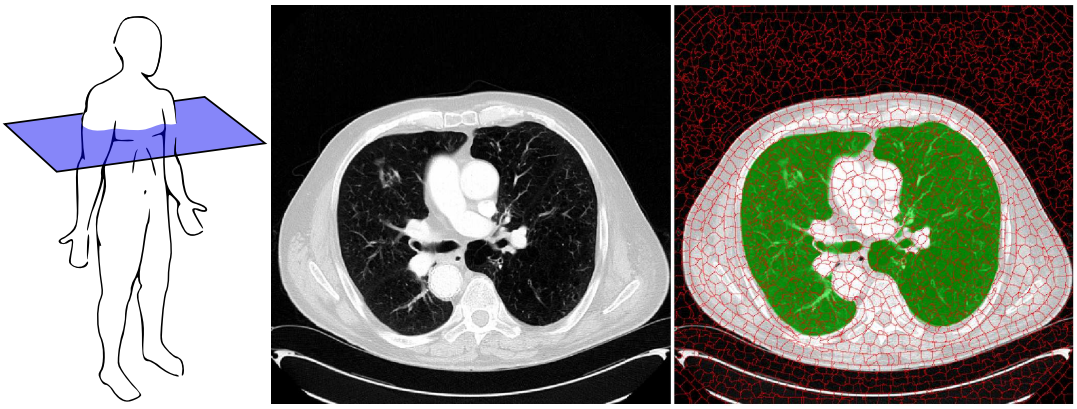


Figure 8.1: A transversal cut of a thorax-CT in the dataset. At the very right side supervoxels are visualized by red borders representing the borders of the supervoxels. The lung mask is indicated by a green overlay.

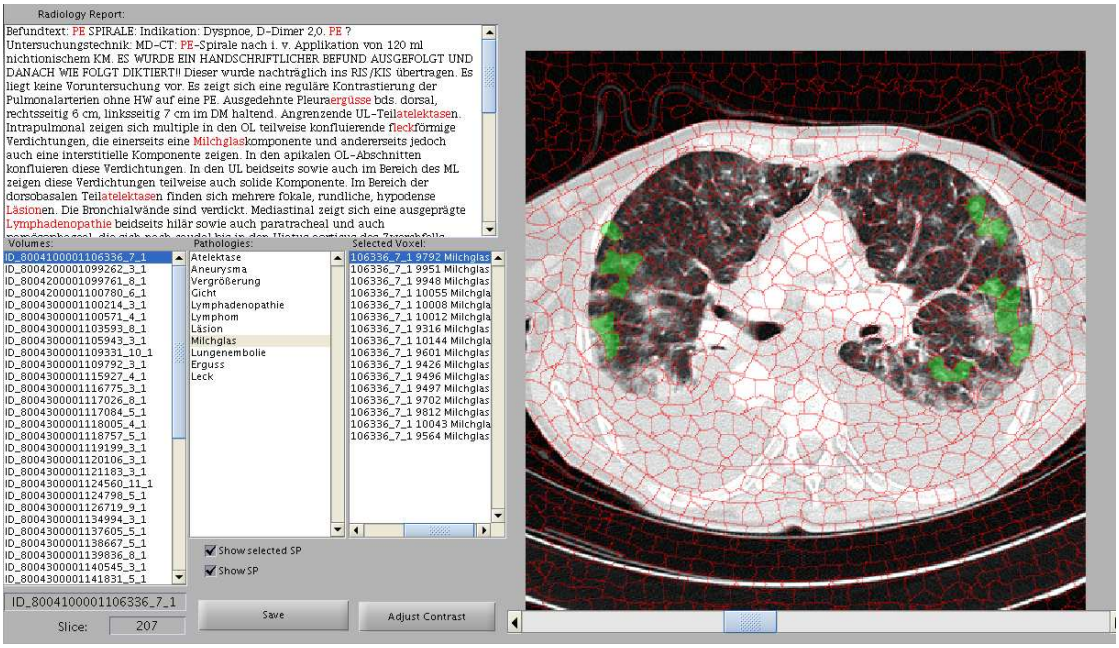
by Andreas Burner and is used for lung segmentation in [18]. Volumes which show an insufficient lung segmentation quality were excluded from the test dataset in advance. Table 8.1 gives an overview on volume-size and number of supervoxels in the dataset. The overall number of supervoxels which are at least partially in the lung is $N = 2\ 220\ 071$.

	min	max	median
Size	$[356 \times 356 \times 264]$	$[635 \times 635 \times 1080]$	$[496 \times 496 \times 443]$
# Supervoxel	18755	106661	36582
# Supervoxel(Lung)	2703	12275	6957

Table 8.1: Minimum, Maximum and Median values for different properties of the 318 thorax CT image dataset. “Size” describes the extend of the volumes in number of isotropic voxels. “# Supervoxel(Lung)” denotes the number of supervoxels that are entirely or at least partially in the lung.

8.2 Information Need and Query Set Extraction

The query-set used is comprised of manually selected regions in the form of supervoxels, extracted from the test collection. To construct the supervoxel based query-set an annotation tool has been developed. The tool does not only provide information about the superpixel oversegmentation of the image but also shows the radiological report and lists the extracted pathologies for a volume. For each of the considered anomalies a representative set of supervoxels is extracted from multiple volumes. Figure 8.2a shows a screenshot of the tool developed for query set construction.



(a)

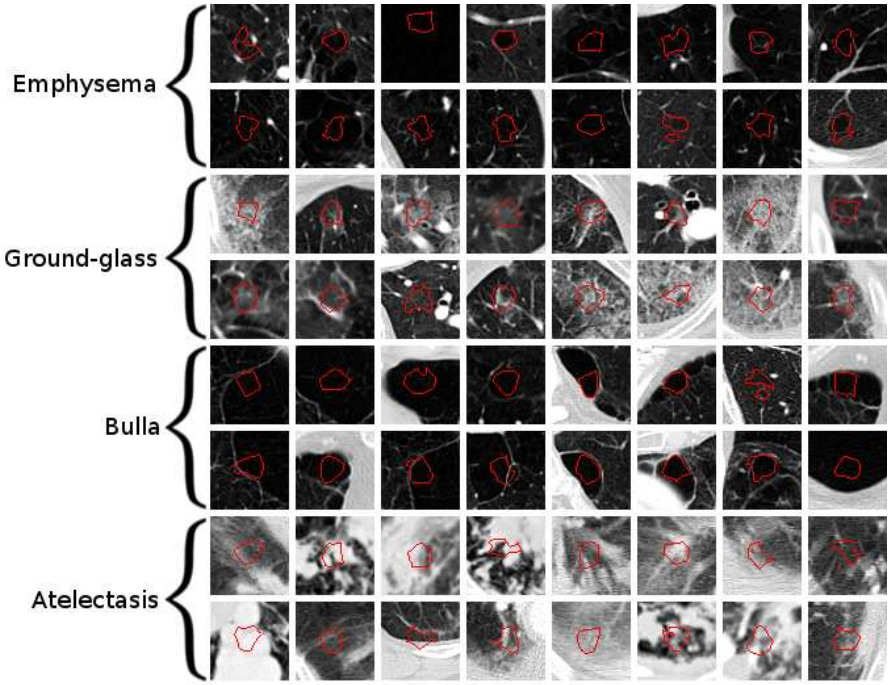
Figure 8.2: The annotation tool used to extract a set of query supervoxels. The image, supervoxels, the radiological report a list of extracted pathology terms associated with the image are provided to the annotator. After selection of a pathology, the annotator can select supervoxels in the image which are associated with the term.

The anomalies considered for retrieval are chosen with respect to: (1) their frequency of occurrence in the dataset and (2) their visual impact in a CT image. The frequency of occurrence is considered in order to avoid a too limited set of relevant volumes, that may be biased to imaging quality or just little variation of appearance. Some diagnostic labels extracted from radiological reports may only be meaningful in a wider context of medical history and diagnosis. Thus, only anomalies, where a direct connection between diagnostic label and visual appearance in the CT image can be drawn are considered. A radiologist suggested Emphysema, Ground-glass, Bulla and Atelectasis for these experiments and the given dataset. Figure 8.4 gives information about the most frequent terms extracted and their frequency in the dataset. Table 8.3 additionally provides the names and RadLex-ID[1] for the terms. Figure 8.3a shows a subset of the queries to illustrate the visual appearance of the anomalies chosen. Table 8.2 lists the number of queries extracted for each anomaly and the number of volumes they have been extracted from.

8.3 Experimental Evaluation in Visual Information Retrieval

The aim of the developed components is a system that is able to rank a set of images for a given query. A query is a set of supervoxels. The search is based on visual similarity and a

[1]RadLex is a unified language of radiology terms, *http://www.radlex.org/*



(a)

Figure 8.3: A subset of the queries for each of the four anomalies used to evaluate the methods. From top to bottom, query supervoxel for *Emphysema*, *Ground-glass*, *Bulla* and *Atelectasis*. The red lines mark the border of the supervoxel.

	# Queries	# Volumes
Emphysema	61	10
Ground-glass	63	6
Bulla	34	6
Atelectasis	39	8
Total	197	30

Table 8.2: The number of queries extracted for each anomaly. The second column shows the number of different volumes of which the queries have been extracted from.

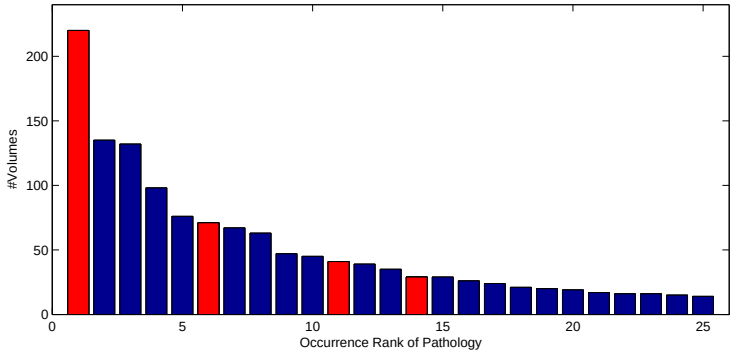


Figure 8.4: The occurrences of the top 25 most frequent pathologies extracted from the radiological reports. The pathology terms selected for evaluation are marked red. Table 8.3 provides information about names and exact number of occurrence.

successful query has to return a set of images that contain one or more regions with the same diagnostic label as the query. This section describes measurements to judge the effectiveness of a CBIR system. The measures used distinguish between two classes of records, relevant and non-relevant.

Two basic notions used in information retrieval are *precision* and *recall* [47].

Precision(P) is the fraction of retrieved records that are relevant [47]

$$Precision = \frac{\#\ of\ relevant\ records\ retrieved}{\#\ of\ retrieved\ records} = P(relevant|retrieved) \tag{8.1}$$

Recall(R) is the fraction of relevant records that are retrieved [47]

$$Recall = \frac{\#\ of\ relevant\ records\ retrieved}{\#of\ relevant\ records} = P(retrieved|relevant) \tag{8.2}$$

The division of the records into two classes also allows to show these terms on the basis of a contingency table [47] (Table 8.4).

$$P = \frac{tp}{tp + fp} \tag{8.3}$$

$$R = \frac{tp}{tp + fn} \tag{8.4}$$

Depending on the system and the task to analyse, one measure may be more important than the other. For an image retrieval system which shows just a very limited number of results, the precision will be of more interest than the recall rate [47]. The given problem is in a ranked retrieval context and sets of retrieved records are naturally given by lists of top k results based

Rank	Occurence	Name(German)	RadLex-ID
1	**220**	**Emphysem(Emphysema)**	**RID4799**
2	135	Zyste	RID3890
3	132	Läsion	RID38780
4	98	Sklerose	RID5227
5	76	Erguss	RID4872
6	**71**	**Atelektase(Atelectasis)**	**RID28493**
7	67	Rundherd	RID29173
8	63	Lungenembolie	RID4834
9	47	Infiltrat	RID39317
10	45	Granulom	RID3953
11	**41**	**Bulla(Bulla)**	**RID28502**
12	39	Bronchiektasie	RID28496
13	35	Lymphadenopathie	RID3798
14	**29**	**Milchglas(Ground-glass)**	**RID45726**
15	29	Metastasierung	RID5231
16	26	Chronisch obstruktive Lungenerkrankung	RID5317
17	24	Raumforderung	RID3874
18	21	Aszites	RID1541
19	20	Hämangiom	RID3969
20	19	Pneumonie	RID5350

Table 8.3: The 20 most frequent terms extracted from the radiological reports. Note that the pathology names are in german. The anomalies selected for evaluation are bold and the english translation is given in parentheses. In addition, the RadLex-IDs are given for the extracted terms. Figure 8.4 illustrates the term occurrences in a bar plot.

	Relevant	Nonrelevant
Retrieved	true positive (**tp**)	false positive(**fp**)
Not retrieved	false negative (**fn**)	true negative (**tn**)

Table 8.4: Contingency table for classes relevant and nonrelevant.

on a score detailing the similarity of each record to the query. One may expect a decrease of the density of relevant records when moving from the top of a ranking downwards. Hull et al. [38] states, that the measurement of the effectiveness of a ranked retrieval can either be done by (1) measuring the density of relevant records at a chosen recall value or (2) at a fixed rank number. We use measures for both of these groups to compare the components of the developed system.

Precision Recall Curve. By varying the number of results k for a query, k sets are generated. For each set, precision and recall values can be plotted, resulting in a *precision-recall curve* [47]. This curve is based on fixed k and the precision value for a fixed recall has to be interpolated by choosing the highest precision found for any recall level $R' \geq R$ [47]. Thus, the

interpolated precision P_{ip} at a fixed recall level R is defined as:

$$P_{ip}(R) = \max_{R' \geq R} P(R') \tag{8.5}$$

The justification for this is, that usually, one would accept the effort to inspect a few more results if it would increase the percentage of relevant records viewed [47]. Based on this definition, the precision can be plotted on a set of standard recall points. Although any range of recall levels would be possible, the de-facto standard became 11 points starting at a recall level of 0% increasing by 10% up to recall=100% [47, 54]. For each of this recall levels, the arithmetic mean of the interpolated precision at that level among each query in the test collection is calculated. The resulting precision-recall curve is denoted as **11-point interpolated average precision** [47].

Mean Average Precision. To examine a precision-recall curve is informative. However, to simplify the comparison it may be desirable to reduce this information down to a single number. One measure which fulfills this requirement is Mean Average Precision (MAP) [47, 54]. It provides a single value to measure the quality across different recall levels. For a single query, the Average Precision (AP) is the average of the precision at the rank positions of each relevant record. More formally:

$$AP = \frac{\sum_r^k (P(r) \times rel(r))}{|R|}, \tag{8.6}$$

where again, k is the number of documents retrieved, r is the rank, $rel(r)$ returns either 0 or 1 depending on relevance of the record at rank r and $P(r)$ is the precision measured at the specified rank. Note, that for every un-retrieved relevant record, the precision is measured as being zero. The MAP is the average of this value over a set of queries:

$$MAP = \frac{\sum_{q \in Q}(AP(q))}{|Q|}. \tag{8.7}$$

The AP approximates the area under the uninterpolated precision-recall curve [47]. So that for the MAP, no fixed recall levels are chosen and no interpolation is performed. Each query is weighted equally in the final result number.

Precision Measured at a Fixed Rank The precision at a fixed rank k is simply defined as the precision for the set of top k records retrieved [54]. This measure is justified for search systems where just a fixed number of results is usually examined by the user (e.g. the first page of results returned by a web search engine). Müller et al. state, that the precision at fixed rank along with MAP are widely used to evaluate systems in IR research [54].

ROC Curve and AUC Also used for evaluation in IR is the Receiver Operating Characteristics (ROC) curve. The curve is a plot of the *true-positive* rate or *sensitivity* against the *false-positive rate* (1-specificity) [28]. Note, that in our case sensitivity is just another word for recall (see Equation 8.4). The false-positive rate is defined as:

$$(1 - specificity) = \frac{fp}{fp + tn} \tag{8.8}$$

Section	Equation	Symbol	Value	Description
6.2.2	6.9	γ	5	Impact factor of voxel intensity in LBP3d/I descriptor
6.2.2		S	400	Vocabulary size for texture bags approach
6.2.2	4.1	L	10	Depth of vocabulary Fern
6.2.2	4.12	sub_l	5	Number of dimensions used on node test of vocabulary fern
6.2.3	6.12	E	4	Number of scales on which texture bag histograms are extracted from
6.2.3	6.12	$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	$4, 3, 2, 1$	Weighting factor for scales in the joint histogram
6.2.4		$d*$	100	Target dimensionality for PCA and RP
6.3.2	6.18	E	1200	Number of ferns in Ensemble
6.3.2	6.19	$L_{\mathcal{E}}$	8	Ferns depth of ferns in $\mathcal{E}$
6.3.3	6.27	K	5000	Number of leaf nodes used for neighborhood approximation

Table 8.5: Parameters set for the experiments conducted.

The diagonal line $(0, 0)$ to $(1, 1)$ in a ROC curve represents a retrieval system which randomly selects records from the database. To elevate the plot from this diagonal onto the upper triangular region, the system has to explore some information in the data [28]. A one-value performance measure is the Area Under the ROC curve (AUC). The AUC is equivalent to the probability, that the retrieval system will rank a randomly chosen positive instance higher than a randomly chosen negative instance [28].

8.4 Experimental Set-Up, Parameter Setting and Initial Experiments

In this section the experimental set-up and the parameter setting used for evaluation is described. Initial experiments are conducted to (1) evaluate properties that do not effect the retrieval quality (e.g. runtime of descriptor extraction) and (2) to define parameters. Subsection 8.4.2 gives information about the parameter setting for the random BoVW approach and analyses the influence of the candidate vocabulary selection procedure on the distribution of word frequencies. Subsection 8.4.3 gives results on runtime performance of a random vocabulary compared to a distance based vocabulary for assigning new candidates to words. Subsection 8.4.4 visualizes some SP features for the parameter setting used. Table 8.5 gives an overview of the parameters.

8.4.1 Hard- and Software Setup

If not differently stated, all experiments have been conducted on a system with two Intel Xeon X5650 processors with 12 cores, 24 threads and 72496MB of RAM. The operating system is Linux 2.6.32 and the Matlab version used is R2011a.

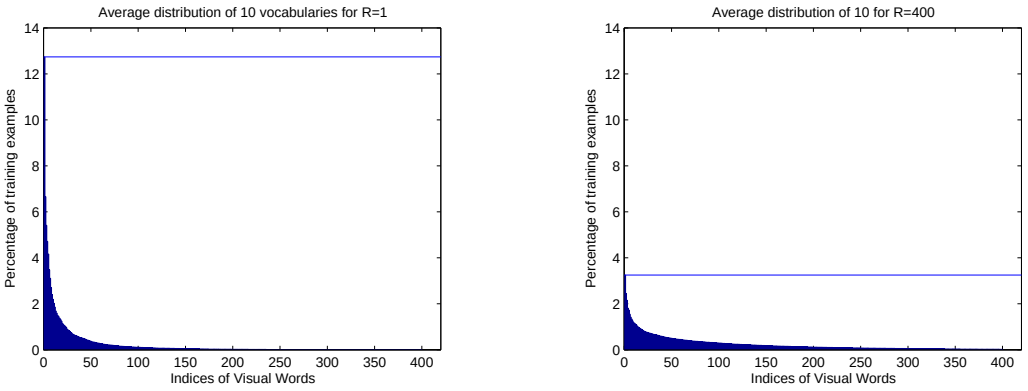


Figure 8.5: Distribution of 10^6 training examples over the vocabulary given in percent for (a) purely random vocabulary without quality assessment by variance and (b) selection of the vocabulary with the lowest variance from $R = 400$ candidates. The horizontal lines denote the share of the word with the highest frequency. Due to the random nature of the approach, the average result of 10 runs is given. The vocabularies in (a) cover in average 99.76% of the training examples and in (b) 97.42% of the examples.

8.4.2 Parameters for BoVW Vocabulary Construction

This subsection describes initial experiments to define parameters of the random vocabulary learning process. To conduct this experiments 10^6 LBP3D/I descriptors are extracted from a set of 40 image out of the dataset described. The target vocabulary size was set to $W = 400$, and fern depth to $L = 10$. These parameters are set with respect to settings reported in literature for comparable methods and applications. Burner et al. used vocabulary size 300 [18] for retrieval of medical images, Mu et al. set $W = 1000$ [52] in his implementation of a random vocabulary. The fern depth L was set by trial and error. With $L = 10$, the fern produces more partitions (1024) than the target vocabulary size but only a small number $(2 - 3\%)$ of non-empty words are trimmed back.

Figure 8.5 illustrates the distribution of features without (a) and with (b) candidate vocabulary selection. Due to the random nature of the approach, 10 vocabularies are constructed for each setting and the average results are shown.

Figure 8.6 shows the effect of parameter R (size of candidate vocabularies) on the word frequency variance. With increasing R, the features are distributed more homogeneous (smaller variance of the word frequencies).

The following parameters have been set heuristically or on the basis of experiments. It is not in the scope of this work to find the optimal parameter setting for the feature extraction approach. Alternative settings may vary in its performance for different anomalies anyway. The weight parameter for voxel intensity γ of the LBP3d/I features is set to 5. Vocabularies are trained on $E = 4$ scales. The vocabulary training is performed on 1.75×10^6 examples of LBP3d/I voxel descriptors for each scale. The scale weighting parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are set to $4, 3, 2, 1$ (The impact of high scale words are elevated). The number of dimensions considered

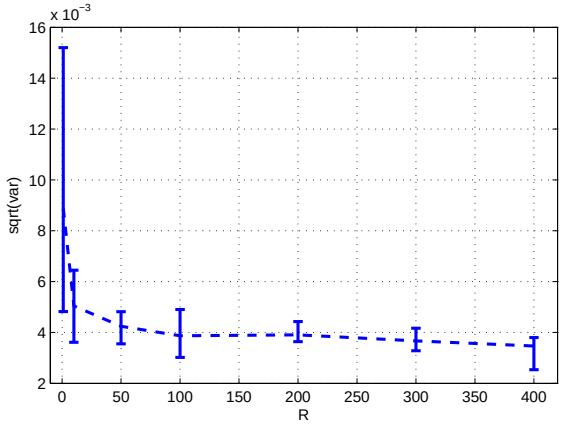


Figure 8.6: The influence of candidate size R on the histogram variance. The results are averaged over 10 runs. The error bars show the minimum and maximum value of the 10 runs.

on RF node tests (4.12) sub_l is set to 5. Based on the given parameter setting, the dimensionality of a texture bag feature vector $\mathbf{f}^{TB}$ is $W \times E = 400 \times 4 = 1600$.

8.4.3 Random Vocabulary Runtime Experiment

In this subsection, evaluation of speed improvement gained by using a random vocabulary compared to a distance based vocabulary is given. For this experiment a volume of size 587x587x547 is used. The lung mask covers $\sim 28 \times 10^6$ voxels. The runtime, needed for assigning the 27 dimensional LBP3d/I descriptors of all voxels to a word of a vocabulary of size 400 is measured. The random vocabulary is given by a fern according to the proposed random vocabulary method and the settings stated in Section 8.4.2. A distance based vocabulary is given by 400 randomly sampled descriptors. The optimized mex C++ implementation for nearest neighbor search provided by the *Yael Toolbox*[2] is used for finding of the closest word. The random vocabulary method is completely implemented in Matlab. The assignment of $\sim 28 \times 10^6$ descriptors to 400 random words needs ~ 6 seconds. The assignment of $\sim 28 \times 10^6$ descriptors to its nearest neighbor out of 400 vectors needs ~ 140 seconds. Thus, a speed improvement of over 95% is achieved by using a random vocabulary.

8.4.4 Parameters and Initial Experiments on Semantic Profile

For training (fern generation and determination of the relative term frequencies in the leafs) of the SP ferns ensemble, all 318 volumes in the dataset are used. We restricted the number of anomaly terms (weak labels) to the 100 most frequent terms as this number determines the target dimensionality of the SP vector. Thus, the resulting f^{SP} descriptor is in $\mathbb{R}^{100}$. The size of the ferns ensemble is set to 1200 and the ferns depth is set to 8. Setting the ferns depth to 8 has the practical reason that it allows to store the leaf index as an unsigned 8-bit integer variable. The

[2] https://gforge.inria.fr/projects/yael/

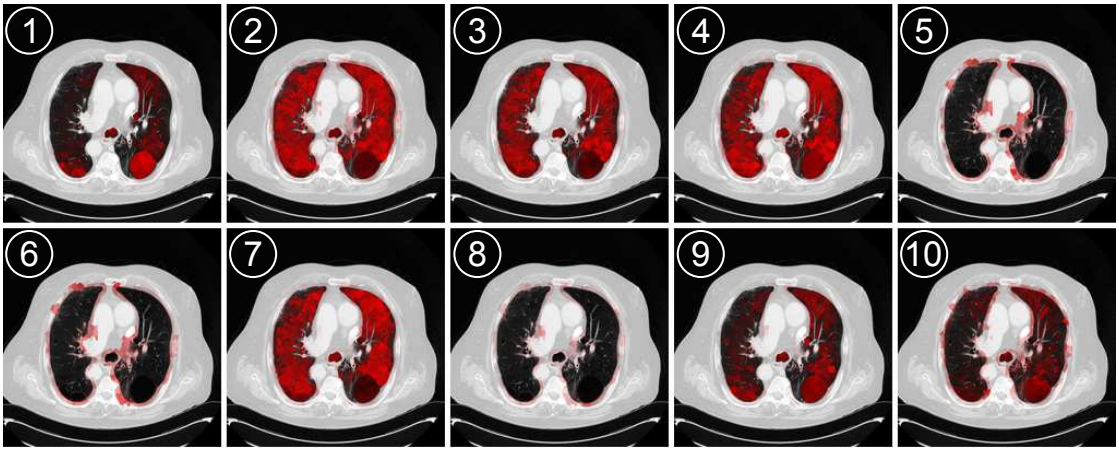


Figure 8.7: Visualization of the first 10 components of the $\mathbf{f}^{SP}$ descriptor. The color intensity reflects the components value of a supervoxel. Note, that due to independent scaling, intensities are not comparable between two components. The terms used to learn the components depicted in this figure can be looked up in Table 8.3.

number K (in Equation 6.27) of top partitions with highest relative class frequency to determine the term indicator is set to 5000.

Figure 8.7 shows the first 10 components of the $\mathbf{f}^{SP}$ descriptor in an axial slice of a volume. The color intensity reflects the impact of the component on a supervoxel. For the sake of presentation, the values within the volume for each component are scaled to $[0, 1]$. Thus, relative comparisons between the components can not be conducted on the basis of this representation.

8.4.5 Visualizing Supervoxel Similarities

This subsection describes a first initial experiment to compare the descriptors $\mathbf{f}^{SP}$,$\mathbf{f}^{PCA}$ and $\mathbf{f}^{RP}$. All descriptors are of dimension 100. To show their general ability to describe the similarity between two supervoxels, the distance of a single query supervoxel to all supervoxels in a volume is visualized. In order to enhance the contrast between similar and non-similar supervoxels a similarity function which scales and inverts the distances in a non-linear way is used. If $\bar{\mathcal{D}}$ is the mean of the set of distances retrieved and d is a single distance measure, the similarity value used is

$$sim(d) = e^{-d^2/\bar{\mathcal{D}}^2}. \tag{8.9}$$

Figure 8.8 illustrates the scaling and the color scheme used to visualize the similarity. Figure 8.9 shows transversal slices of a volume and the similarity encoded as color. A query supervoxel from the “Bulla” query-set is chosen. The columns show different transversal slices of the volume, the rows are the results using different descriptors. Figures showing the similarities in volumes for queries representing other anomalies are attached in Appendix A.1.1.

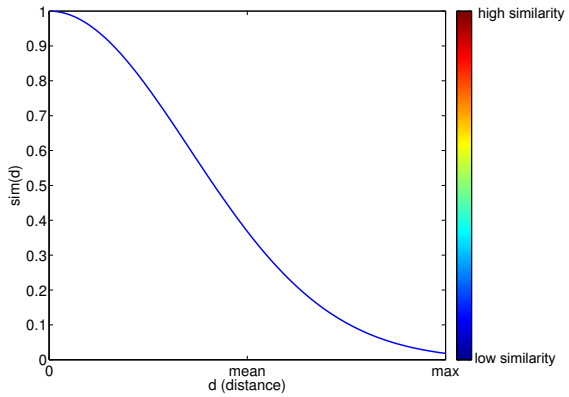


Figure 8.8: Color scheme used for visualization of supervoxel similarity. The similarity between two supervoxels is calculated by Equation 8.9.

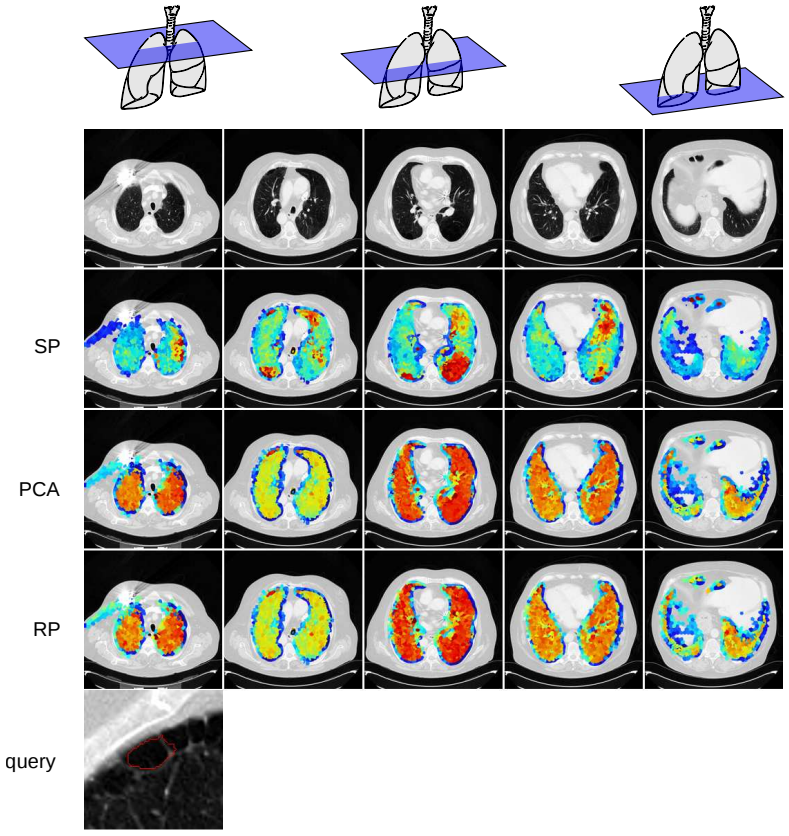


Figure 8.9: Visualization of similarities between the supervoxels in the volume and a distinct query supervoxel. In this case, the voxel is chosen from the set of query voxels associated with the anomaly “Bulla”. The columns are transversal slices through the thorax CT. The rows represent the result for different feature vectors.

8.4.6 Parameter Setting and Implementation of Indexing Methods

This subsection describes parameter settings and implementation details for the indexing methods analysed.

PQ and IVFPQ

The methods IVFPQ and PQ are trained on a set of 70 000 descriptors randomly sampled from $\mathcal{N}^{SP}$. The number of subquantizer centroids $C*$ is set to 8 for all experiments. That allows to store a subquantizers response in an unsigned 8 byte integer variable. The number of subquantizers m is thus set to the desired code length in byte. For the IVFPQ method, the number of cluster centers of the coarse quantizer C' is set to 200.

A version of the publicly available source code[3] of the PQ and IVFPQ method is used to conduct the experiments. The implementation is in Matlab. However, the actual distance computation by summing up the precalculated distances to the subquantizers centroids (Equation 5.19 and 5.16) is implemented in C.

SH

For the experiments conducted, SH is trained on 70 000 descriptors. No other parameter than the desired code length is needed for SH. The publicly available Matlab code[4] is used to generate the binary codes. However, a publicly available C++ implementation[5] of Norouzi et al. [57] is used to compute the Hamming distance between the codes.

LSHE

No training is needed for LSHE and the code computation is performed on all vectors at one time. No parameters except the desired target code length is needed for this method. We implemented LSHE in Matlab. The Hamming distance computation is performed by the same implementation used for the SH codes.

RF

To use RF for indexing, the method described in Section 7.5 is implemented. $R = 1200$ ferns of depth $s = 8$ are trained on $100\ 000$ training descriptors randomly sampled from $\mathcal{N}^{SP}$. The implementation is entirely in Matlab. The Hamming distance for retrieval is computed by the same algorithm and implementation as used for SH and LSHE codes.

8.4.7 Experimental Set-Up for Index Comparison

To evaluate the ANN methods used the approximated kNN results are compared to the exact kNN according to the exact Euclidean distance. A retrieved entry is considered relevant, if

[3] http://people.rennes.inria.fr/Herve.Jegou/projects/ann.html

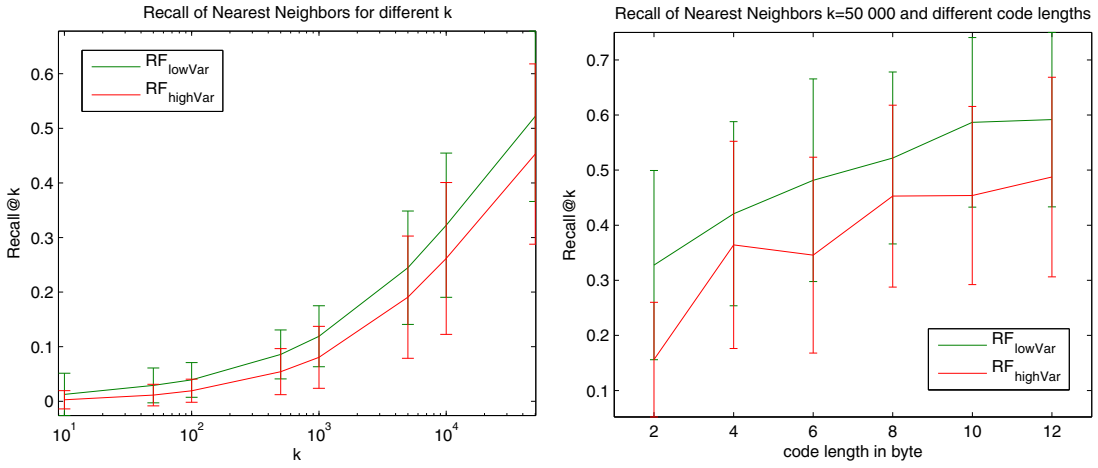
[4] http://www.cs.huji.ac.il/ yweiss/SpectralHashing/

[5] https://github.com/marklar/min-loss-hashing/blob/master/matlab/utils/hammingDist2.cpp

it is also represented in the exact nearest neighbor result retrieved by exact calculation of the distances. As done in other works which analyse ANN methods [48, 41], the *recall* according to this definition is calculated for a certain rank (result list length). The aim of this measure is to show the approximation quality of the ranking rather than the distance approximation. The $\mathbf{f}^{SP}$ image descriptors of the 197 manually selected supervoxels (Table 8.2) are used as queries. The set of $N = 2\ 220\ 071$ SP supervoxel descriptors is used as the database. The recall on a certain rank is the average of the 197 results.

8.4.8 Experiments on RF for Indexing

The idea to use RF is described in Section 7.5. The variance of example frequencies in the generated partitions is used as quality criteria for a code sequence. In this subsection, the validity of this approach is analysed. For each of the 1200 8-bit code sequences the variance of the code frequencies (example frequencies in the partitions) is calculated. Subsequently, for a certain target code length, two groups of ferns are chosen. One group (*lowVar*) is comprised of the ferns that generate the lowest variances among the 1200 candidate ferns and the second group (*highVar*) is formed by the ferns that generate the highest variances. The experimental setup corresponds to the setup described in Subsection 8.4.7. Comparison is done by recall of exact nearest neighbors. Figure 8.10a shows the average recall rates for different ranks using 8 byte codes. Figure 8.10b shows recall rates for rank 50 000 and for different code lengths.



(a) (b)

Figure 8.10: Comparison of codes that are comprised of sequences with low and high variation of representatives. The recall of nearest neighbors according to exact Euclidean distance is used as criteria. Experimental setup (queries and database) as described in Section 8.4.7 is used. The errorbars show the standard deviation. (a) Recall rates for codes of length 8 byte and ranks from 10 to 50 000. (b) Recall rates for rank 50 000 and different code lengths.

CHAPTER 9

Experiments and Results

This chapter describes the experiments conducted and shows the results. Section 9.1 describes comparison of supervoxel descriptors and 9.2 the comparison of ANN methods.

9.1 Comparison of Supervoxel Descriptors

In this section, retrieval results using the descriptors $\mathbf{f}^{PCA}$, $\mathbf{f}^{RP}$ and $\mathbf{f}^{SP}$ are shown. The aim of this evaluation is to show and compare their ability to describe visual similarity with respect to radiological relevance and to visualize retrieval results. This is done by using the data, queries and measures described in Chapter 8. In the following, the abbreviations PCA, RP and SP are used whenever one of the respective descriptor is used.

9.1.1 Experiments

The comparison of the descriptors is performed by ranking on the supervoxel and the image level. The measures used are described in the following:

Ranking on supervoxel level:

Due to the lack of ground truth labels for supervoxels, the weak image labeling is used to judge relevance in supervoxel-rankings. A result supervoxel is considered relevant if it originates from a volume carrying the same weak label as the label of the query supervoxel. More formally, if $p_{i,s}$ is a result supervoxel and q a query with known anomaly then

$$rel(p_{i,s}) = \begin{cases} 1 & \text{if } l(q) \in \mathcal{T}_i \\ 0 & \text{if } l(q) \notin \mathcal{T}_i \end{cases} \tag{9.1}$$

The precision gives information about the descriptors ability to describe anomaly specific visual characteristics. Supervoxels belonging to the same volume as the query are omitted in the search.

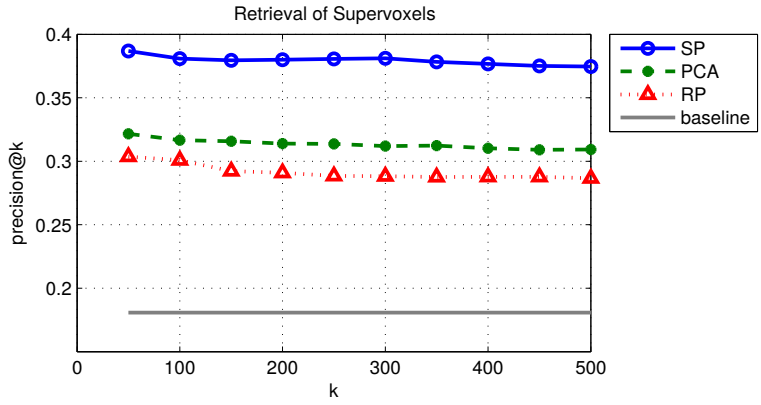


Figure 9.1: Average precision at fixed ranks for different descriptors using the 39 queries in the Atelectasis query-set. The baseline is the share of supervoxels in the database with the weak label associated with Atelectasis. The ordinate indicates the share of supervoxels belonging to a volume with the label Atelectasis.

The evaluation is performed for each of the four anomalies separately. The precision values at the ranks are averaged over the queries.

Ranking on volume level:

In the second part of quantitative evaluation, volume ranking based on a set of query supervoxels is analysed. The labeling of the volumes is used to judge a relevance of the results. To simulate a realistic scenario in which a search is performed by a query region larger than a single supervoxel, sets of 10 queries are used to retrieve a single ranking by using Equation 6.30. The query-set is randomly sampled from the query supervoxels of one anomaly. To produce a stable result, the sampling and the search was repeated 70 times and the average results are reported. The ranking is performed on ranked lists of 60000 nearest neighbors per query-supervoxel according to Euclidean distance. To analyse the performance by looking at the complete result list, the ROC and the 11-point Interpolated Average Precision curve are used.

9.1.2 Results

Figure 9.1 shows the average precision measures at fixed ranks from 50 to 500. All 39 entries in the query set of Atelectasis are used as queries for exact k-NN search. The precision values at the ranks have been averaged over the queries. Supervoxels belonging to the same volume as the query are omitted.

Figure 9.2 shows average precision ranks for fixed k from $5 \leq 100$ by using supervoxels from the Atelectasis query-set and Figure 9.3 shows the ROC and the 11-point Interpolated Average Precision curve .

In this section, plots for Atelectasis queries are presented only. Results for Emphysema, Bulla and Ground-glass are attached in Appendix A.2. To give an overview of the quantitative

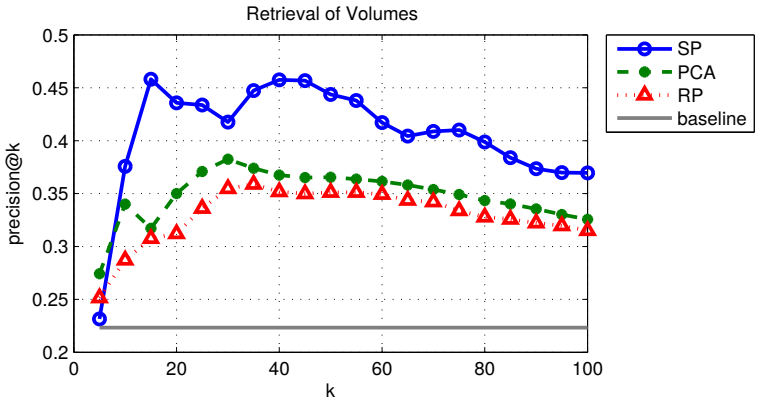


Figure 9.2: Average precision at fixed ranks after performing volume ranking on the basis of different descriptors using 70 query sets of size 5-20 randomly sampled from the queries in the Atelectasis group. The baseline is the share of volumes in the database with Atelectasis label.

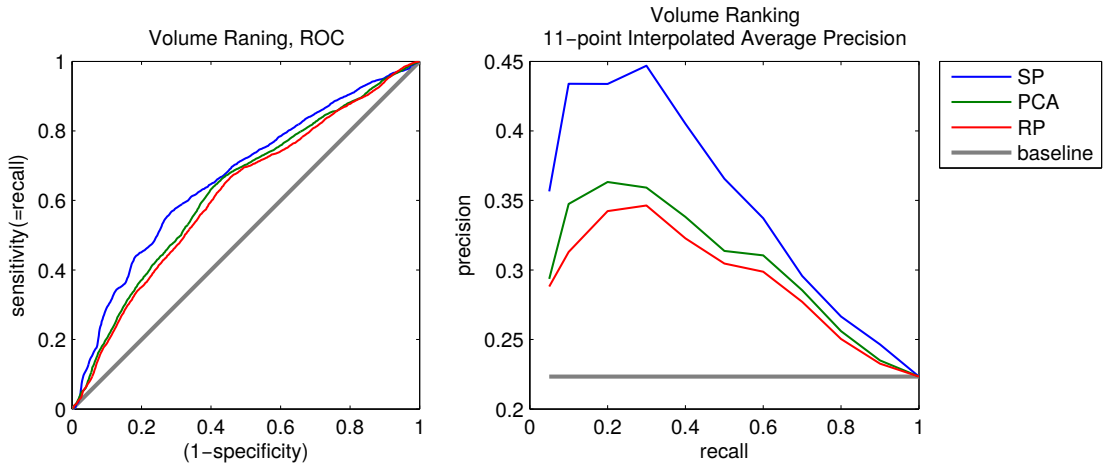


Figure 9.3: ROC (left) and 11-point Interpolated Precision (right) after performing volume ranking on the basis of different descriptors using 70 query sets of size 5-20 randomly sampled from the Atelectasis queries.

results, sets of one-value measures for each of the different query groups are listed in Table 9.1.2. The value PR_{SV}@500 is the average precision at rank 500 for supervoxel retrieval results, PR@10 and PR@30 are the precision values at rank 10 and 30 of the volume retrieval result lists.

Emphysema							
	PR_{SV}@500	AUC	MAP	PR@10	PR@30	bl	bl_{SV}
SP	0.900	0.664	0.825	**1.000**	0.985		
PCA	0.911	0.725	0.863	**1.000**	**1.000**	0.69	0.73
RP	**0.916**	**0.749**	**0.876**	0.997	0.997		

Ground-glass							
	PR_{SV}@500	AUC	MAP	PR@10	PR@30	bl	bl_{SV}
SP	**0.266**	**0.693**	**0.251**	**0.370**	**0.214**		
PCA	0.194	0.620	0.187	0.266	0.168	0.09	0.08
RP	0.169	0.628	0.187	0.270	0.176		

Bulla							
	PR_{SV}@500	AUC	MAP	PR@10	PR@30	bl	bl_{SV}
SP	**0.291**	0.588	**0.194**	**0.349**	**0.172**		
PCA	0.272	0.601	0.177	0.111	0.169	0.13	0.18
RP	0.237	**0.622**	0.177	0.050	0.168		

Atelectasis							
	PR_{SV}@500	AUC	MAP	PR@10	PR@30	bl	bl_{SV}
SP	**0.375**	**0.667**	**0.360**	**0.376**	**0.418**		
PCA	0.309	0.632	0.320	0.340	0.382	0.22	0.14
RP	0.287	0.618	0.308	0.287	0.355		

Table 9.1: Multiple one-value measures for Emphysema, Ground-glass, Bulla and Atelectasis. PR_{SV}@500 is precision at rank 500 for supervoxel retrieval results, PR@10 and PR@30 are the precision values at rank 10 and 30 for volume retrieval results. bl is the share of volumes in the dataset holding the relevant label and bl_{SV} is the share of supervoxels holding the relevant weak label.

9.1.3 Visual Results

In this subsection, visual retrieval results for supervoxel ranking and volume ranking are shown. In order to show the visual similarity between a query and its nearest neighbors in the database the 4 most similar supervoxels among all supervoxels in the database are shown. Supervoxels with the same volume of origin as the query are omitted. To allow a fair comparison but make

a compact representation of the results possible, the best (highest number relevant anomalies) results for each method are shown only. To visualize a supervoxel, the transversal slice with the largest cross-section of the supervoxel is used. The first columns (indicated by green boxes) show the query supervoxels.

Figure 9.5 shows the top 5 ranked volumes by using the set of Bulla queries. The ranking is performed by using all queries of an anomaly for the volume ranking metric (Equation 6.30). In Appendix A.1.2, retrieval results of the top 5 ranked volumes for the other three query sets are attached.

9.1.4 Discussion

In this section, three, dimensionality reduced representatives of the texture bag descriptor $\mathbf{f}^{TB}$ are evaluated and compared. All three descriptors show recall rates higher than the baseline. The one-value measures listed in Table 9.1.2 show better results for three of the four analysed anomalies when the SP descriptor is used. Visual results show top ranked supervoxels and volumes. It is not in the scope of this work to interpret the radiological relevance of this results.

9.2 Comparison of Indexing Methods

In this section, comparison of the ANN search methods PQ, IVFPQ, RF, SH and LSHE is performed. Parameter setting and implementation of the methods is described in Subsection 8.4.6. The results show (1) comparison regarding the approximation of the nearest neighbors result list and (2) the influence of the approximation on the ranking of relevant supervoxels and volumes.

9.2.1 Experiments

To study the kNN approximation, the set-up described in Section 8.4.7 is used. The approximation quality of the methods is compared for different result list lengths and varying code-lengths.

In addition, the influence of approximation on the ranking of relevant supervoxels and volumes is studied. The comparison is performed in the same way as the comparison of the descriptors (Section 9.1.1). However, this evaluation is performed for Ground-glass queries only. For ranking on supervoxel level the precision is plotted for different ranks. For ranking of volumes, the precision@k, ROC curve and 11-point Interpolated Average Precision is plotted.

To visualize the effect of approximation on the similarity measure, a volume slice with an overlay of approximated supervoxel similarities to a query is shown and the highest ranked Ground-glass volumes using the indexing methods are shown.

9.2.2 Results

Figure 9.6 shows recall rates for varying values of the result length k. 64-bit (8 byte) codes are used for approximation. In the case of LSHE, a 4 byte single precision value is used to store the vectors length and 4 bytes are used to approximate the angle.

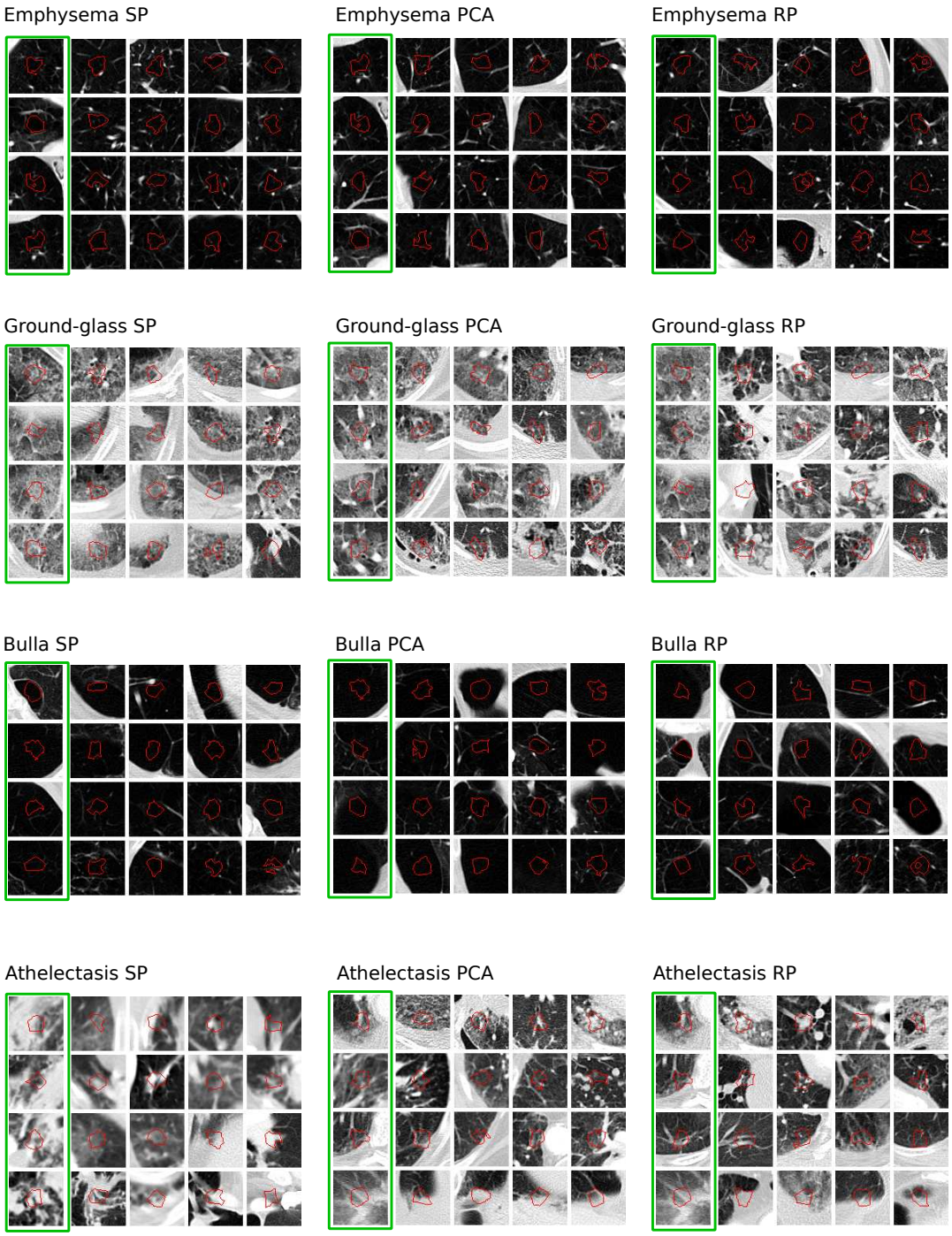


Figure 9.4: Visualization of nearest neighbor supervoxel in the database for different query supervoxel, anomaly and descriptor. Queries are depicted in the first column and are marked by a green box. 4 nearest neighbors per query supervoxel are shown. The search is performed exhaustive on the complete database except the supervoxels belonging to the same volume as the query.

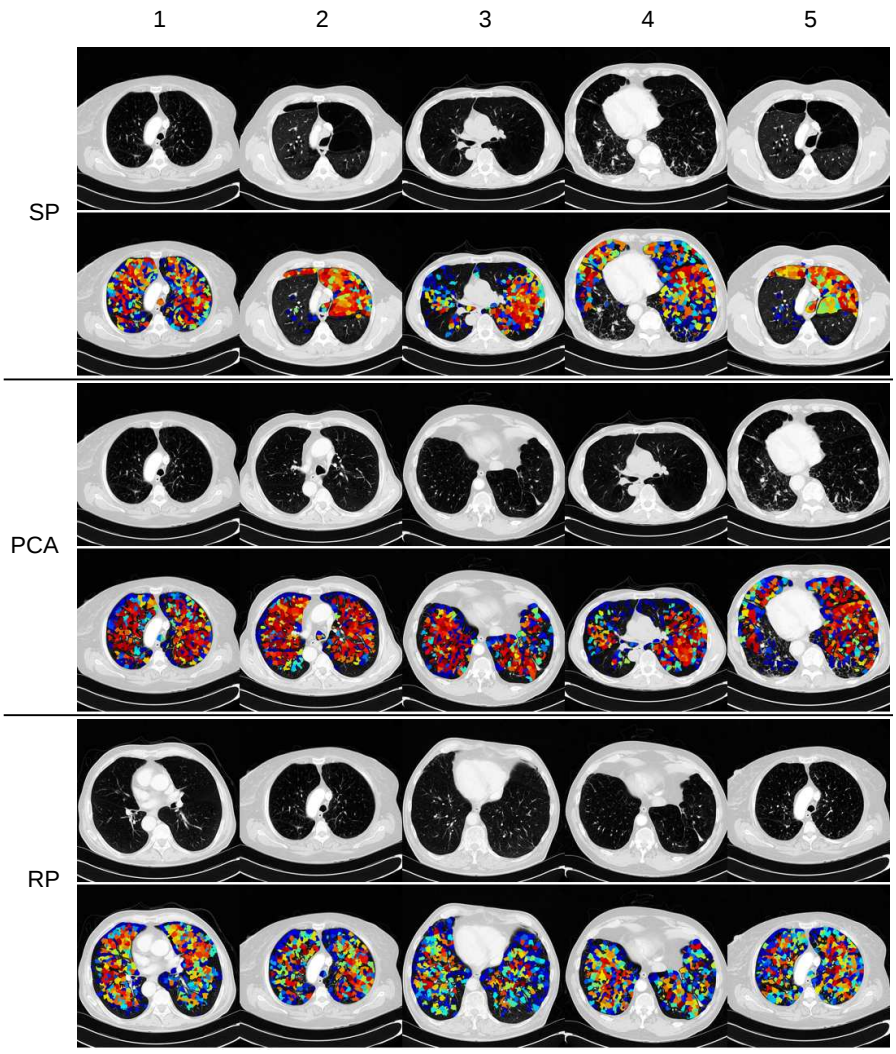


Figure 9.5: Examples of retrieved volumes by using the set of Bulla query supervoxels. The five top ranked volumes using different descriptors are shown. The columns show the ranks, and the rows the different descriptors used. The slice with the highest sum of similarities is depicted for each volume. The slices are shown with and without a similarity overlay.

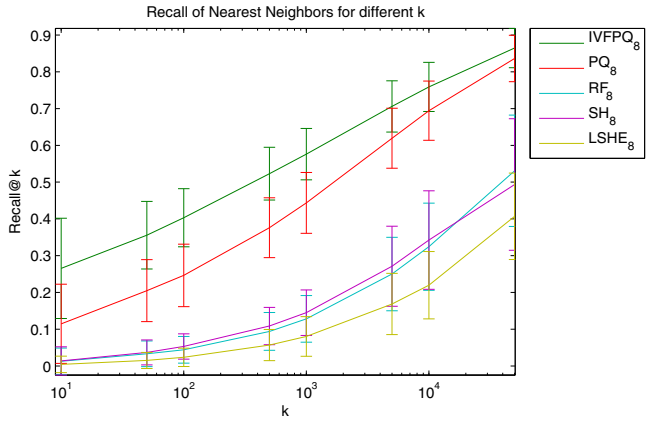


Figure 9.6: Recall rates for varying values of k for $10, 50, 100, 500, 1\text{k}, 5\text{k}, 10\text{k}$ and 50k. Comparison of the approaches IVFPQ, PQ, RF, SH and LSHE. The $\mathcal{N}^{SP}$ dataset is used as database. The depicted recall rates are the average of results for 197 queries. The error bars show the standard deviation.

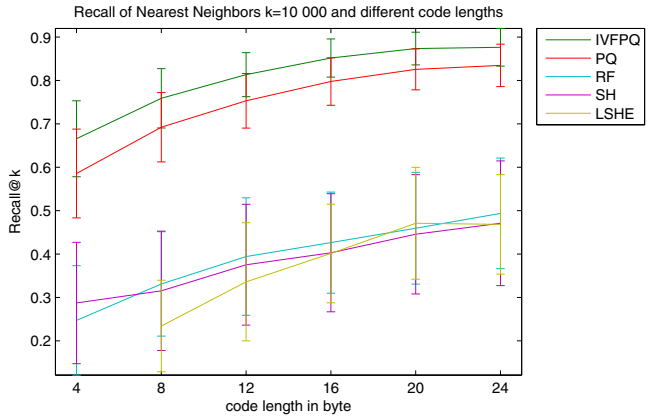


Figure 9.7: Recall rates for $k = 10\text{k}$ and varying code lengths from 4 to 24 bytes. Comparison of the approaches IVFPQ, PQ, RF, SH and LSHE. The depicted recall rates are the average of results for 197 queries. The error bars show the standard deviation of these results. No result for LSHE and code length 4 bytes is given, as 4 bytes are needed to store the vector length.

Figure 9.6 shows recall rates for a fixed $k = 10\text{k}$ and varying code lengths. No result is given for LSHE and code length 4 bytes as 4 bytes are needed to store the vector length and no bits are left to approximate the angle.

Figure 9.8 shows precision rates for supervoxel based retrieval using the queries associated with the Ground-glass anomaly. Figure 9.9 shows results for volume ranking based on the approximated distances. Table 9.2.2 shows one value measures for voxel and volume ranking using 8-byte codes.

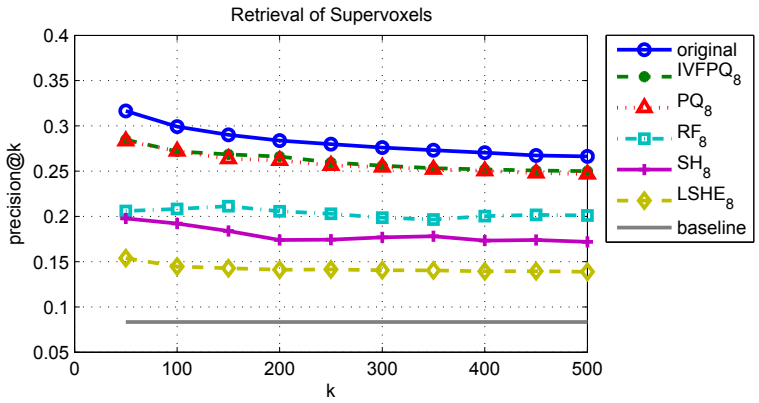


Figure 9.8: Approximated retrieval results for *ground-glass* queries. Comparison of IVFPQ, PQ, RF, SH and LSHE with original SP vectors. Average result of 63 queries.

Ground-glass					
	PR_V@500	AUC	MAP	PR@10	PR@30
original	0.266	0.693	0.251	0.370	0.214
$IVFPQ_8$	0.250	0.692	0.245	0.354	0.210
PQ_8	0.247	0.694	0.245	0.363	0.203
RF_8	0.201	0.689	0.242	0.390	0.193
SH_8	0.172	0.666	0.233	0.384	0.228
$LSHE_8$	0.139	0.685	0.218	0.331	0.191

Table 9.2: One-value measures for ranked lists using Ground-glass queries. PR_{SV}@500 is precision at rank 500 for supervoxel ranking, PR@10 and PR@30 are the precision values at rank 10 and 30 for volume ranking.

9.2.3 Visual Results

Figure 9.10 shows the approximation of the similarities of supervoxels in a slice to a singe query. Depicted are the results for different methods and different code lengths from 4 to 24 byte.

Figure 9.11 visualizes retrieval results by using all representatives of the Ground-glass anomaly in the query set. The result using exact nearest neighbor search, IVFPQ and RF is shown. Results for PQ, LSHE, SH and RF are attached in Appendix A.3.1.

9.2.4 Discussion

Six methods to approximate nearest neighbor search have been analysed on a set of SP supervoxel descriptors. The experiments are based on comparison to exact calculation of nearest

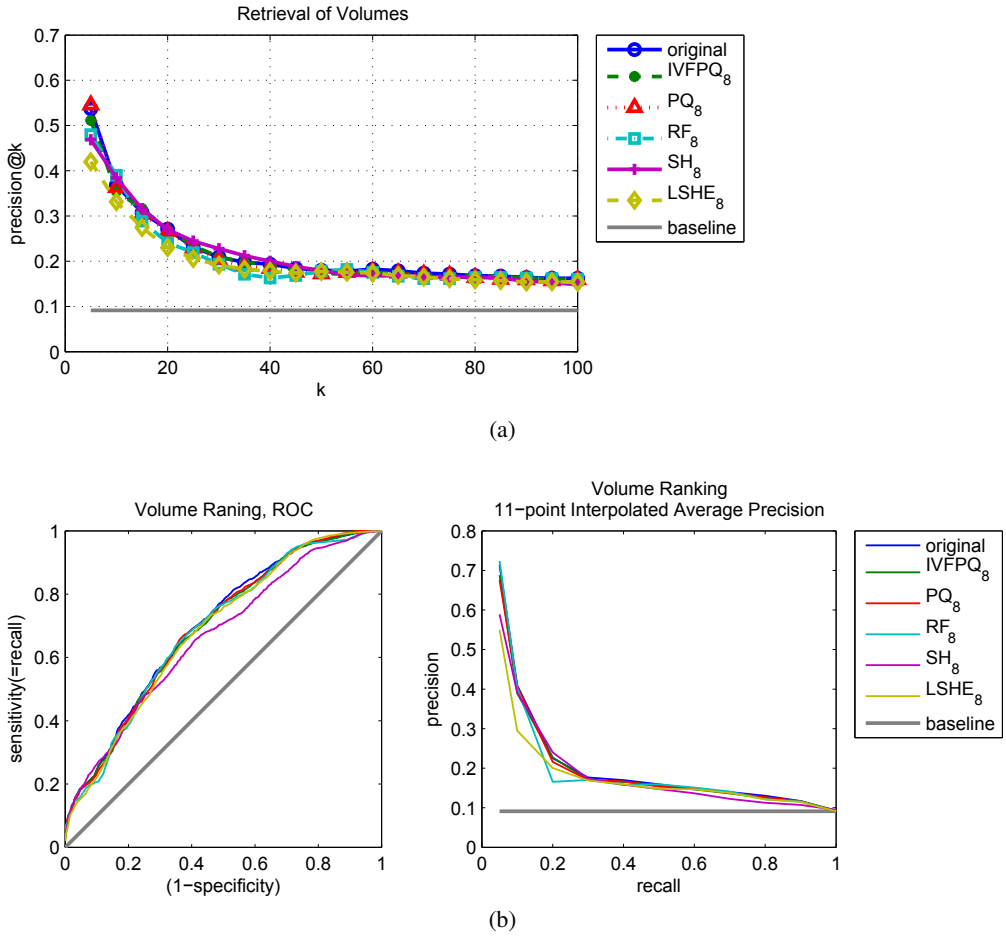


Figure 9.9: Comparison of approximated supervoxel retrieval and volume ranking. (a) Precision at ranks from 5 to 100 and (b) ROC curve and 11-point Interpolated Average Precision for the ranking of all volumes. 8-byte code length for all encoding techniques is used.

neighbors according to Euclidean distance. For ranks from 10 to 50 000, the exhaustive and the non-exhaustive PQ techniques show better recall rates compared to the binary encoding techniques. IVFPQ, the non-exhaustive version of PQ shows better recall for all ranks but also for code lengths from 4 to 24 bytes. The results reported by Marukatat et al. [48] depicted in Figure 5.10 can not be confirmed for the dataset used. Murakatat reports better or equal recall rates for LSHE and SH when compared to PQ and code lengths between 4 and 16 bytes are used. Reasons for the diverging results may be the discrepancy of the dataset size, dimensionality or the structure of the data. Murakatat uses a database of 60 000 while the conducted experiments are performed on a database of size larger than 2M. LSHE needs 4 bytes (when single precision is used) to store the vector length needed to reconstruct an approximated Euclidean distance. This may explain the inferior nearest neighbor approximation for code lengths smaller than 12 byte (Figure 9.7). However, with increasing code length, the recall reaches the level of the other

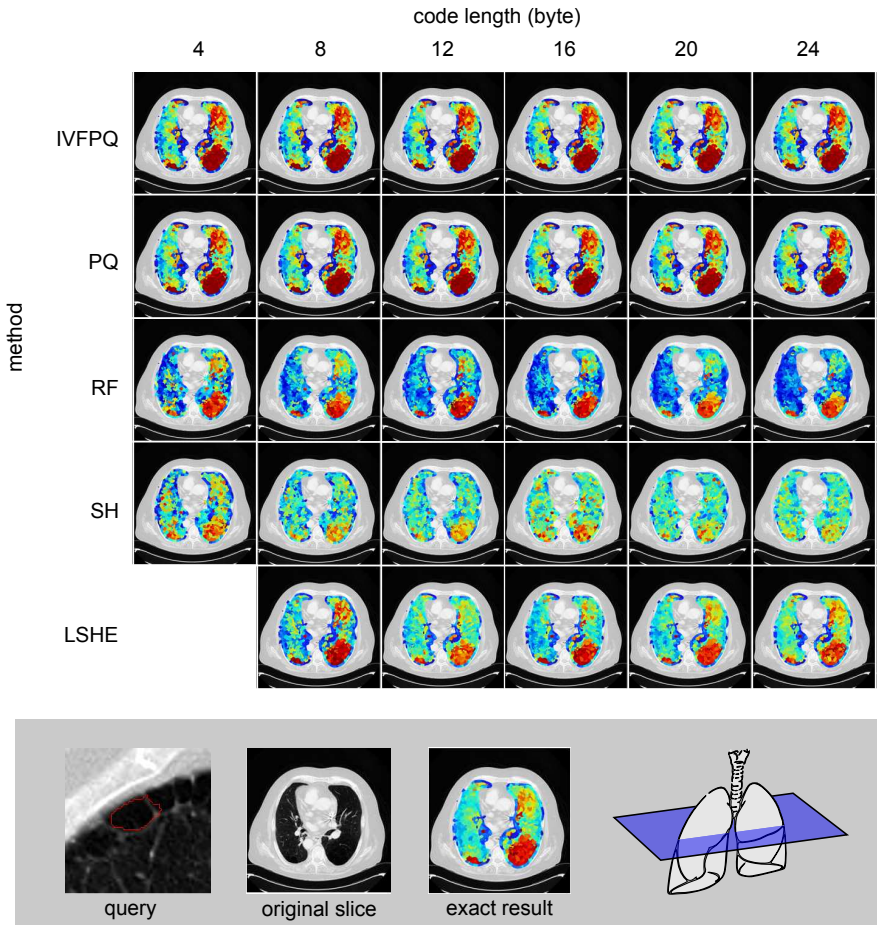


Figure 9.10: Visualisation of supervoxel similarities to a single query in a transerval slice for different approximation techniques and code lengths. The query and the exact result are depicted at the bottom. Color scheme according to Equation 8.9.

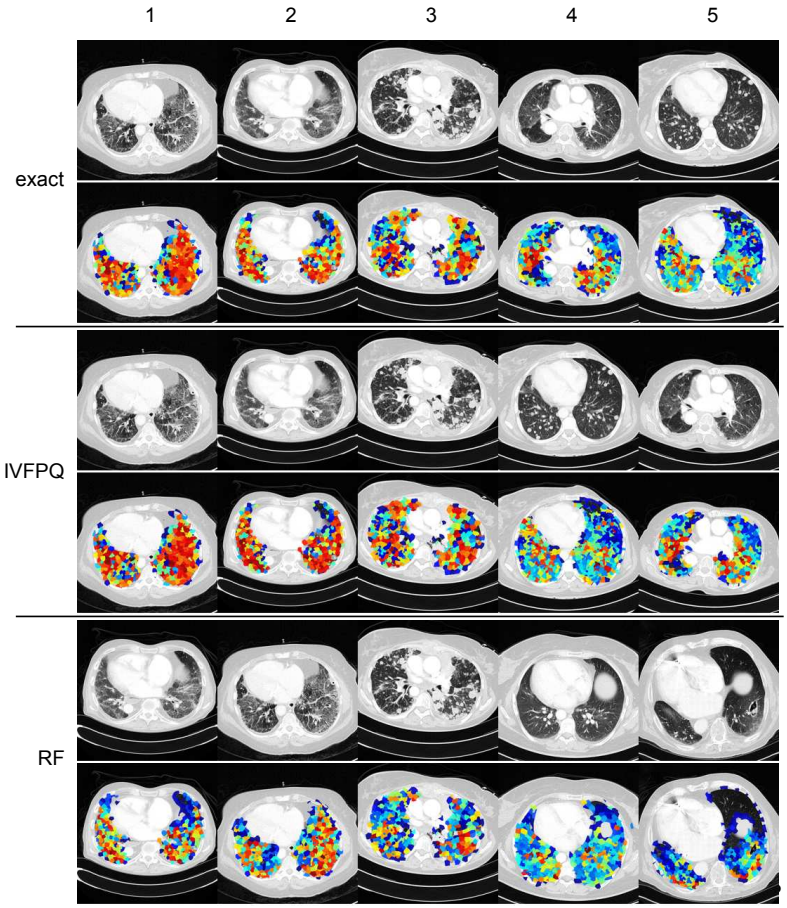


Figure 9.11: Volume ranking by using all queries representing the Ground-glass anomaly. Comparison of retrieval result between the exact Euclidean distance, the approximated Euclidean distance by IVFPQ and RF as a representative of search in hamming space. 8-byte encoding of SP image descriptors is used to generate the approximate results.

binary encoding techniques RF and SH. Even with comparable recall rates of the binary encoding techniques, the visual similarity is approximated differently as can be seen in Figure 9.10. As expected, the methods approximating the Euclidean distance show higher resemblance to the exact results. Evaluations for supervoxel anomaly retrieval reflect the results for nearest neighbor lists (9.8). The Volume ranking metric seems to tolerate approximations in the supervoxel ranking lists as can be seen in Figure 9.9 and Table 9.2.2. Visual results of volume ranking given for 8 byte codes and Ground-glass anomaly show that the ranking changes due to approximated supervoxel distances (Figure 9.11). Considering 8 byte codes, the memory needed for the index is reduced by 98% (from 840MB to 17MB) exclusive overhead such as PQ centroids.

CHAPTER 10

Conclusion

In this final chapter, a summary of the entire work and ideas for further research is provided.

Section 10.1 summarizes background, problems, solutions, evaluation and results. Section 10.2 concludes this work with suggestions for further work and improvements of the developed components of a texture based CBIR system for three dimensional medical images.

10.1 Summary

CBIR serves as the context for this work. To give a comprehensive introduction, Chapter 2 covers the topic of CBIR with focus on the medical domain.

Chapter 3 explains a methodology and design how three dimensional medical images can be represented for CBIR. Oversegmentation (partitioning of the image into supervoxels), LBP (voxel feature vector) and BoVW (supervoxel feature vector) on multiple scales are the basic concepts that serve as starting point for this thesis. However, several limitations of this approach are identified:

1. The assignment of voxel features to visual words takes 95% of the texture bag feature extraction process. This is caused by the high number ($28 * 10^9$ per volume) of voxels due to three dimensions of medical images.

2. The number of supervoxels (7 000 per volume) and the dimensionality of the feature vectors (1200) require suitable indexing in order to meet speed requirements of interactive systems. Furthermore, the memory footprint of a supervoxel descriptor is identified as an issue. The memory needed to describe a supervoxel limits the number of entries to be loaded in a certain time and stored in main memory (in case the index is stored on secondary memory).

3. No semantic information is considered in the definition of the descriptor. Textual information attached to radiological images is identified as information input for learning techniques to diminish the semantic gap.

This work shows a way to overcome this limitations.

The concept of RF is explained in detail in Chapter 4. RF is used in three different parts of the work: (1) to generate a random vocabulary as an alternative to distance based vocabularies, (2) acts as the core components of the weakly supervised leaning technique developed and (3) is used to generate binary codes used for approximate nearest neighbor retrieval.

Chapter 5 gives an extensive description of indexing methods reported in literature. Exact and approximate nearest neighbor techniques are described in detail. ANN is identified as suitable and promising for the data given. A comparison of encoding techniques for ANN is conducted on the basis of results reported in literature.

Chapter 6 describes the supervoxel descriptor developed. The voxel descriptor LBP3d/I, BoVW and a random vocabulary is used to generate a high dimensional supervoxel descriptor. With respect to subsequent indexing, PCA and RP are used as two alternatives to reduce the dimensionality of the vectors. In addition, textual information in the form of pathology labels is processed by means of weakly supervised learning. This method is used to generate a lower dimensional representation of the texture descriptor called SP. SP describes a supervoxel by a sequence of indicators that reflect a supervoxels coherence to certain pathology labels. In addition, a way to rank images on the basis of supervoxel distances is presented.

Chapter 7 describes the application of the ANN indexing methods IVFPQ, PQ, SH and LSHE on image descriptors and the idea to use RF to generate binary codes for nearest neighbor approximation is presented.

Chapter 8 describes the experimental setup and the data used to evaluate and compare the developed and implemented methods. 318 thorax CT images recorded in the daily routine of a hospital are used to test the CBIR framework. The experiments have been restricted to lungs only. Four anomalies are chosen to form exemplary information need. Initial experiments to find parameters are performed. The improved runtime properties in assigning voxel descriptors to words of a random vocabulary compared to a distance based vocabulary is shown. For the test data used, about 95% (from 140 to 6 seconds) speed improvement is achieved.

Chapter 9 shows results for the ranking experiments. Three supervoxel descriptors are compared. PCA and RP represent the descriptors solely learned on image information. SP involves textual information in the descriptors definition. The results show, that SP allows better ranking for three of the four anomalies used as queries.

The comparison of the ANN methods studied show superior results of the PQ methods when applied on the data used. The approximated ranking of relevant supervoxels is studied. The experiments conducted indicate, that image ranking tolerates a certain degree of approximation in the preceding supervoxel ranking without reducing the retrieval quality considerably.

10.2 Future Work and Improvements

In the following, future work and improvements for each of the components developed are discussed separately.

Texture Bag and Random Vocabulary: The improved runtime properties of a random vocabulary has been shown by experiments. However, the influence of such a vocabulary on the

descriptors quality (ability to capture relevant visual features) has not been studied yet. The influence of different parameters (e.g. vocabulary size, number of scales, intensity weighting, etc.) on the descriptors ability to capture visual features of different anomalies could be analysed and studied.

SP: The SP method is novel and only studied for the given setup and data. Evaluation and testing on different data could concretise the potential of the method. SP was applied on texture bag histograms. However, the application of SP on alternative texture or visual descriptors is possible.

Indexing: The indexing methods used approximate nearest neighbors according to Euclidean distance. The Euclidean distance is a heuristic metric when used for image retrieval. The potential of alternative metrics to improve the retrieval quality could be studied.

Evaluation: The evaluation of the developed CBIR components is conducted on real world data. The assessment of the visual retrieval results by an expert could substantiate the quantitative results reported in this work.

Bibliography

[1] CeyhunBurak Akgül, DanielL. Rubin, Sandy Napel, ChristopherF. Beaulieu, Hayit Greenspan, and Burak Acar. Content-based image retrieval in radiology: Current status and future directions. *Journal of Digital Imaging*, 24(2):208–222, 2011.

[2] Mohamed Aly, Mario Munich, and Pietro Perona. Indexing in large scale image collections: Scaling properties and benchmark. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 418–425. IEEE, 2011.

[3] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468. IEEE, 2006.

[4] Artem Babenko and Victor Lempitsky. The inverted multi-index. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3069–3076. IEEE, 2012.

[5] Bahman Bahmani, Ashish Goel, and Rajendra Shinde. Efficient distributed locality sensitive hashing. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2174–2178. ACM, 2012.

[6] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. *The R*-tree: an efficient and robust access method for points and rectangles*, volume 19. ACM, 1990.

[7] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1000–1006. IEEE, 1997.

[8] Richard Bellman, Richard Ernest Bellman, Richard Ernest Bellman, and Richard Ernest Bellman. *Adaptive control processes: a guided tour*, volume 4. Princeton university press Princeton, 1961.

[9] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[10] Stefan Berchtold, Christian Böhm, Daniel A Keim, and Hans-Peter Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 78–86. ACM, 1997.

[11] Stefan Berchtold, Daniel A Keim, and Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. *Readings in multimedia computing and networking*, page 451, 2001.

[12] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Database Theory—ICDT'99*, pages 217–235. Springer, 1999.

[13] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[14] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[15] Christian Böhm, Stefan Berchtold, and Daniel A Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, 33(3):322–373, 2001.

[16] Henrik Bostrom. Estimating class probabilities in random forests. In *Sixth International Conference on Machine Learning and Applications*, pages 211–216. IEEE, 2007.

[17] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[18] Andreas Burner, René Donner, Marius Mayerhoefer, Markus Holzer, Franz Kainberger, and Georg Langs. Texture bags: anomaly retrieval in medical images based on local 3d-texture similarity. In *Medical Content-Based Retrieval for Clinical Decision Support*, pages 116–127. Springer, 2012.

[19] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(8):1026–1038, 2002.

[20] Vittorio Castelli and Lawrence D Bergman. *Image databases*. Wiley Online Library, 2002.

[21] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *Int J Math Models Meth Appl Sci.*, 1(4):300–307, 2007.

[22] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[23] Ricardo da Silva Torres and Alexandre Xavier Falcão. Content-based image retrieval: Theory and applications. *Journal of Theoretical and Applied Informatics (RITA)*, 13(2):161–185, 2006.

[24] Pragya A Dang, Mannudeep K Kalra, Thomas J Schultz, Steven A Graham, and Keith J Dreyer. Informatics in radiology render: An online searchable radiology study repository. *Radiographics*, 29(5):1233–1246, 2009.

[25] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[26] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):5, 2008.

[27] Philip Palin Dendy and Brian Heaton. *Physics for diagnostic radiology*. CRC Press, 2011.

[28] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[29] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[30] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, 100(7):750–753, 1975.

[31] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases*, pages 518–529, 1999.

[32] Antonin Guttman. *R-trees: A dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[33] Murat Hamit, Jingjing Zhou, Chuanbo Yan, Li Li, Jianjun Chen, Yanting Hu, and Dewei Kong. Feature extraction and analysis on ct image of xinjiang local liver hydatid by using gray-scale histograms. *Keji Daobao/ Science & Technology Review*, 30(6):66–70, 2012.

[34] Robert Hecht-Nielsen. Context vectors: general purpose approximate meaning representations self-organized from raw data. *Computational intelligence: Imitating life*, pages 43–56, 1994.

[35] Gisli R Hjaltason and Hanan Samet. Ranking in spatial databases. In *Advances in Spatial Databases*, pages 83–95. Springer, 1995.

[36] M. Holzer and R. Donner. Over-segmentation of 3d medical image volumes based on monogenic cues. In *Proceedings of the 19th CVWW*, pages 35–42, 2014.

[37] Xian-Sheng Hua, Linjun Yang, Jingdong Wang, Jing Wang, Ming Ye, Kuansan Wang, Yong Rui, and Jin Li. Clickage: Towards bridging semantic and intent gaps via mining click logs of search engines. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 243–252. ACM, 2013.

[38] David Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338. ACM, 1993.

[39] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[40] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Computer Vision–ECCV 2008*, pages 304–317. Springer, 2008.

[41] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[42] Avinash C Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*, volume 33. Siam, 1988.

[43] Willi A Kalender. *Computed tomography*. Wiley.com, 2011.

[44] Christel Le Bozec, Eric Zapletal, Marie-Christine Jaulent, Didier Heudes, and Patrice Degoulet. Towards content-based image retrieval in a his-integrated pacs. In *Proceedings of the AMIA Symposium*, page 477. American Medical Informatics Association, 2000.

[45] Haibin Ling and Kazunori Okada. Diffusion distance for histogram comparison. In *Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 246–253. IEEE, 2006.

[46] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.

[47] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[48] Sanparith Marukatat and Ithipan Methasate. Fast nearest neighbor retrieval using randomized binary codes and approximate euclidean distance. *Pattern Recognition Letters*, 34(9):1101 – 1107, 2013.

[49] Yoshihiro Mitani, Yusuke Fujita, Naofumi Matsunaga, and Yoshihiko Hamamoto. The use of a slice feature vector of classifying diffuse lung opacities in high-resolution computed tomography images. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on*, pages 60–63. IEEE, 2012.

[50] Andrew Moore. *Efficient memory-based learning for robot control*. PhD thesis, University of Cambridge, Computer Laboratory, 1990.

[51] Frank Moosmann, Bill Triggs, Frederic Jurie, et al. Fast discriminative visual codebooks using randomized clustering forests. *Advances in Neural Information Processing Systems 19*, pages 985–992, 2007.

[52] Yadong Mu, Ju Sun, Tony X Han, Loong-Fah Cheong, and Shuicheng Yan. Randomized locality sensitive vocabularies for bag-of-features model. In *Computer Vision–ECCV 2010*, pages 748–761. Springer, 2010.

[53] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[54] Henning Müller, Paul Clough, Thomas Deselaers, and Barbara Caputo. *ImageCLEF: Experimental Evaluation in Visual Information Retrieval*, volume 32. Springer, 2010.

[55] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. A review of content-based image retrieval systems in medical applications—clinical benefits and future directions. *International journal of medical informatics*, 73(1):1–23, 2004.

[56] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168. IEEE, 2006.

[57] Mohammad Norouzi and David M Blei. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 353–360, 2011.

[58] Timo Ojala, Matti Pietikainen, and David Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, volume 1, pages 582–585. IEEE, 1994.

[59] OpenCV. http://www.opencv.org/doc/tutorials/features2d/feature_flann_matcher/-feature_flann_matcher.html. Accessed: 2013-08-21.

[60] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010.

[61] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *Conference on Computer Vision and Pattern Recognition, CVPR'07.*, pages 1–8. IEEE, 2007.

[62] Olivier Pauly, Diana Mateus, and Nassir Navab. Building implicit dictionaries based on extreme random clustering for modality recognition. In *Medical Content-Based Retrieval for Clinical Decision Support*, pages 47–57. Springer, 2012.

[63] Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. In *Computer Vision–ECCV 2010*, pages 749–762. Springer, 2010.

[64] Foster Provost and Pedro Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[65] GP Robinson, HD Tagare, JS Duncan, and CC Jaffe. Medical image collection indexing: Shape-based retrieval using kd-trees. *Computerized Medical Imaging and Graphics*, 20(4):209–217, 1996.

[66] Geoffrey D. Rubin, John K. Lyo, David S. Paik, Anthony J. Sherbondy, Lawrence C. Chow, Ann N. Leung, Robert Mindelzun, Pamela K. Schraedley-Desmond, Steven E. Zinck, David P. Naidich, and Sandy Napel. Pulmonary nodules on multi–detector row ct scans: Performance comparison of radiologists and computer-aided detection. *Radiology*, 234(1):274–283, 2005. PMID: 15537839.

[67] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[68] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of 13th International Conference on Very Large Data Bases*, pages 507–518. VLDB endowments, 1987.

[69] Jonathon Shlens. A tutorial on principal component analysis. *Systems Neurobiology Laboratory, University of California at San Diego*, 2005.

[70] R Short, Keinosuke Fukunaga, et al. The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory*, 27(5):622–627, 1981.

[71] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[72] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[73] Usha Sinha and Hooshang Kangarloo. Principal component analysis for content-based image retrieval1. *Radiographics*, 22(5):1271–1289, 2002.

[74] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Ninth IEEE International Conference on Computer Vision*, pages 1470–1477. IEEE, 2003.

[75] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[76] McG Squire, Henning Muller, Wolfgang Muller, et al. Improving response time by search pruning in a content-based image retrieval system, using inverted file techniques. In *IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL)*, pages 45–49. IEEE, 1999.

[77] Vildana Sulic, Janez Perš, Matej Kristan, and Stanislav Kovacic. Efficient dimensionality reduction using random projection. In *Computer Vision Winter Workshop*, pages 28–36, 2010.

[78] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *Computational Intelligence and Bioinspired Systems*, pages 758–770. Springer, 2005.

[79] James Z Wang, Nozha Boujemaa, Alberto Del Bimbo, Donald Geman, Alexander G Hauptmann, and Jelena Tesić. Diversity in multimedia information retrieval research. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 5–12. ACM, 2006.

[80] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the international conference on very large data bases (VLDB)*, pages 194–205, 1998.

[81] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753 – 1760. NIPS, 2008.

[82] Horst Wildenauer, Branislav Mičušík, and Markus Vincze. Efficient texture representation using multi-scale regions. In *Computer Vision–ACCV 2007*, pages 65–74. Springer, 2007.

[83] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.

APPENDIX A

Appendix

A.1 Quantitative Results, Descriptors

A.1.1 Distance in Volume

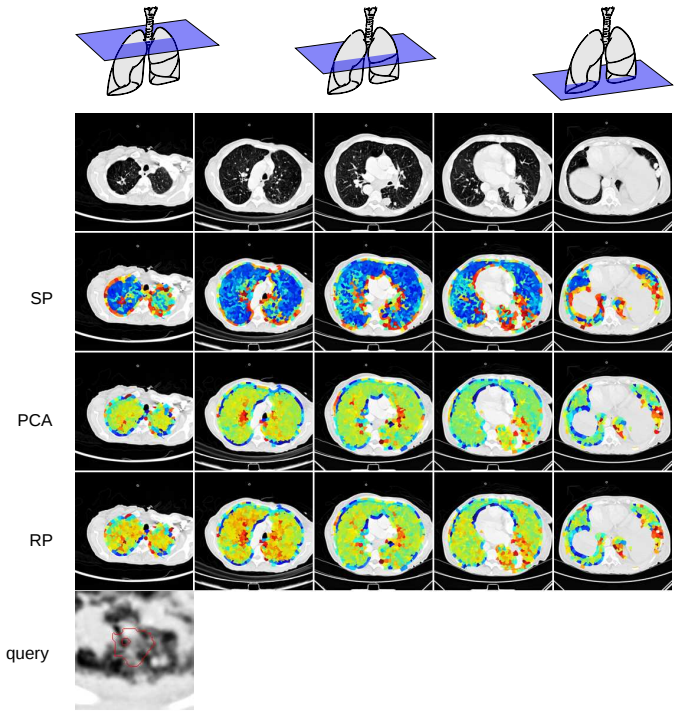


Figure A.1: In Volume Similarity for query voxel representing Athelectasis

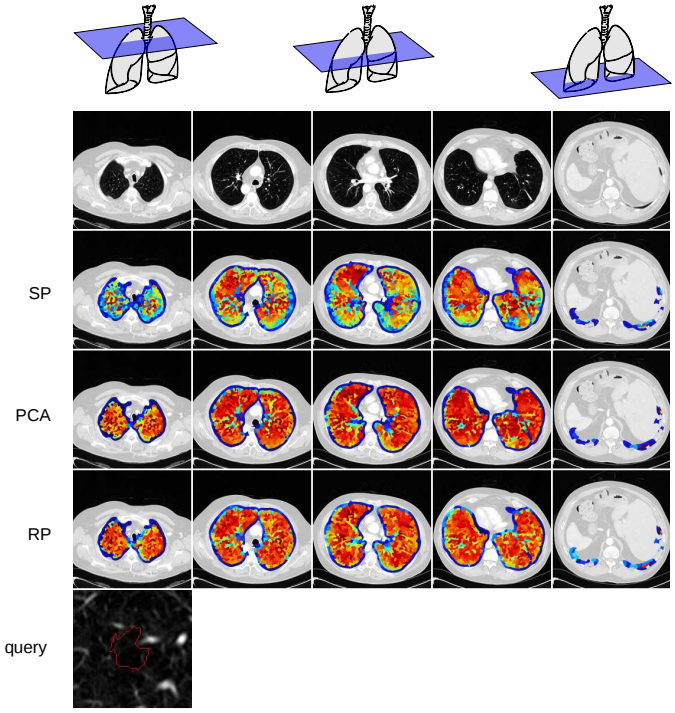


Figure A.2: In Volume Similarity for query voxel representing Emphysema

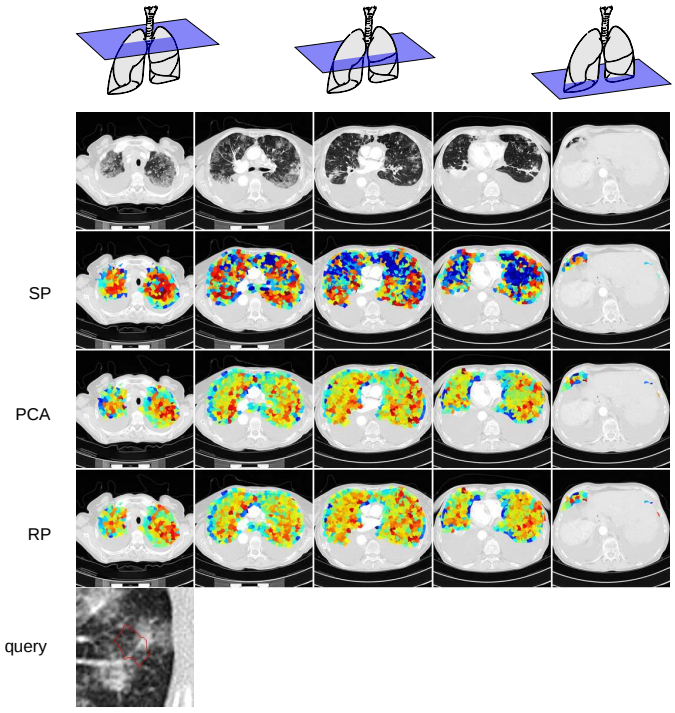


Figure A.3: In Volume Similarity for query voxel representing Groundglass

A.1.2 Retrieval Volume Ranking

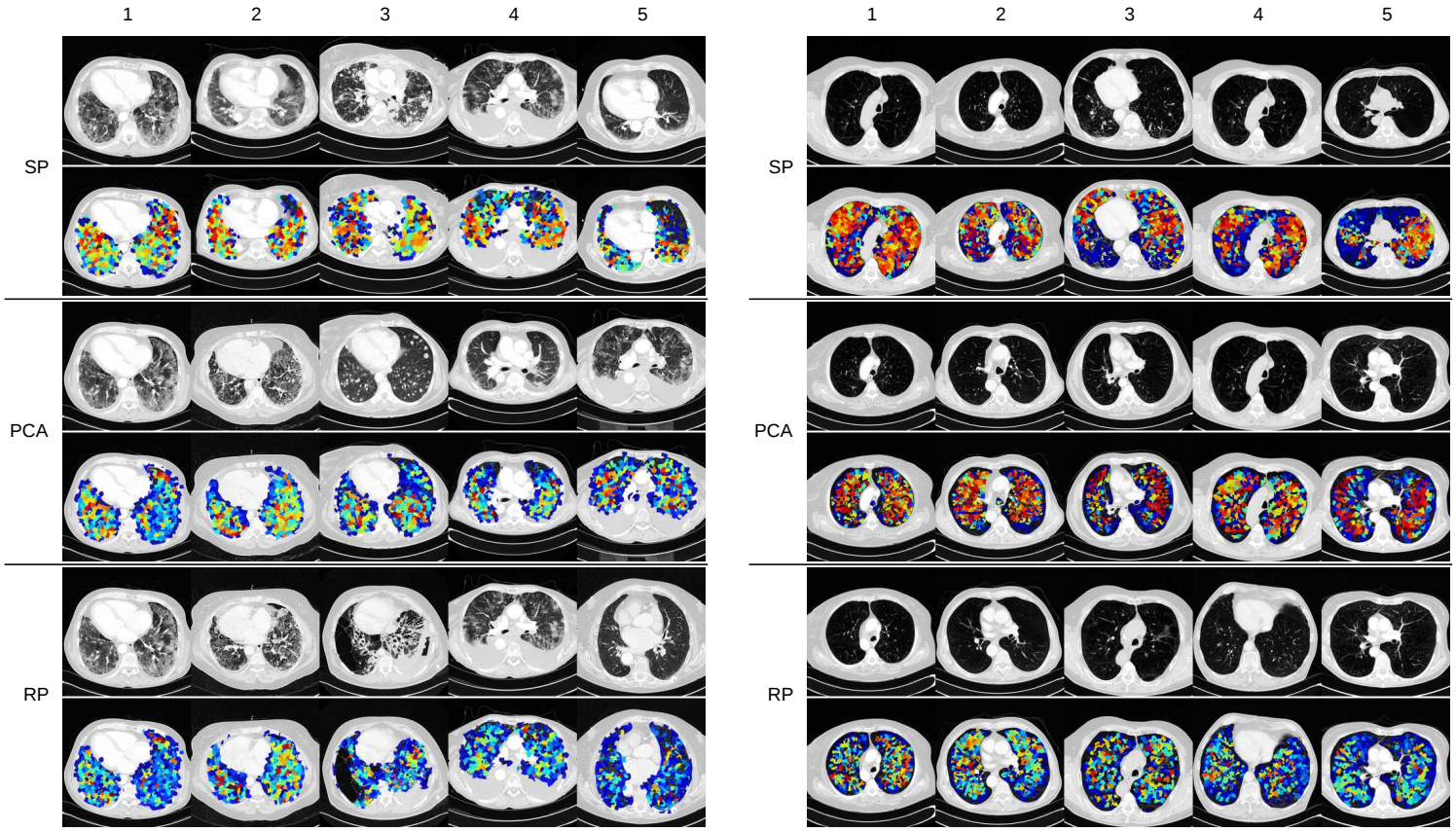


Figure A.4: Top 5 Retrieval result for anomalies Atelectasis (left) and Emphysema (right)

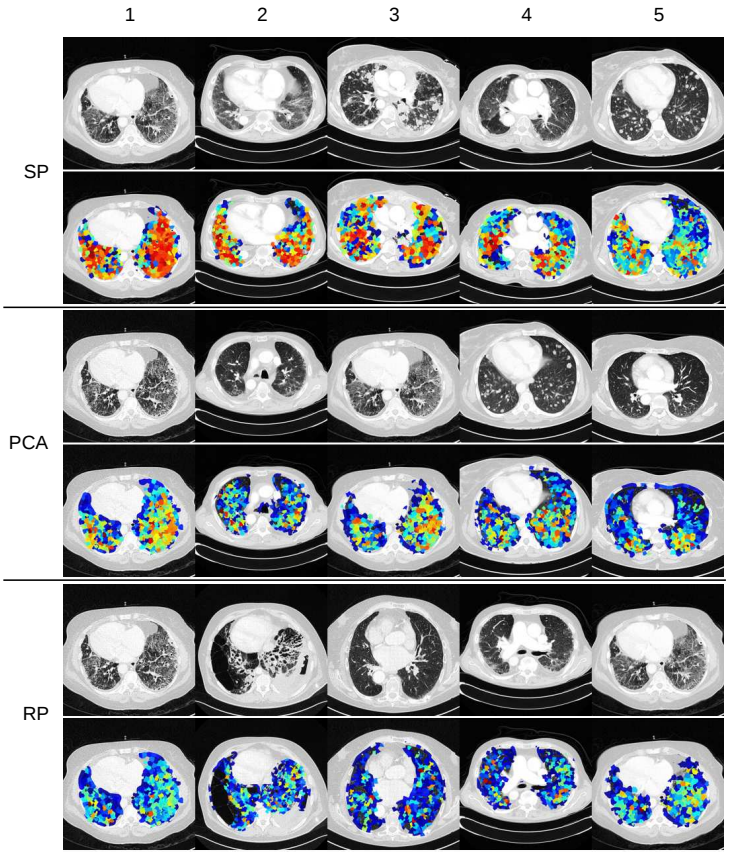
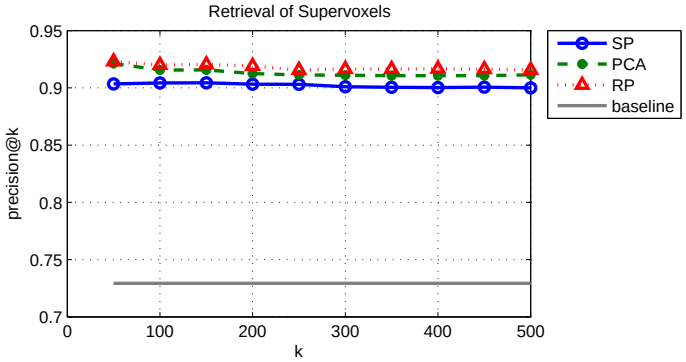


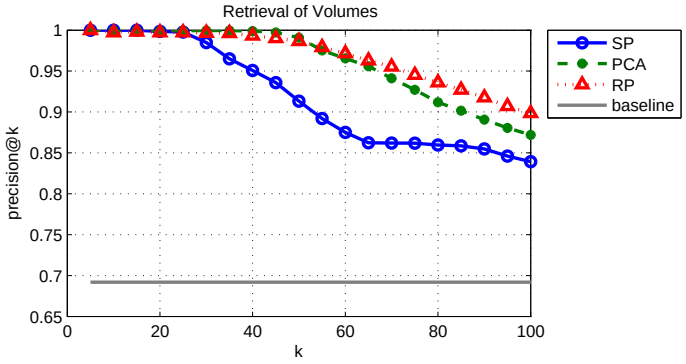
Figure A.5: Top 5 Retrieval result for anomaly Groundglass

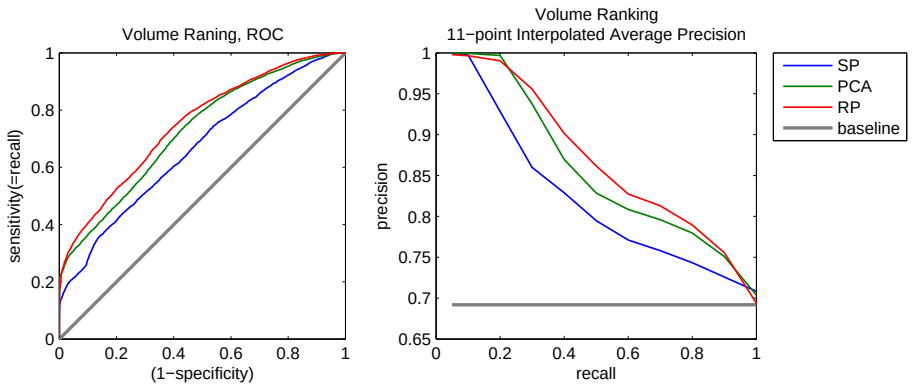
A.2 Quantitative Results on Retrieval for Different Descriptors

Quantitative retrieval results for Emphysema

Anomaly: Emphysema
Number of Queries: 61
Number of Supervoxels: 2220071
Number of Volumes (incl. query volume): 318

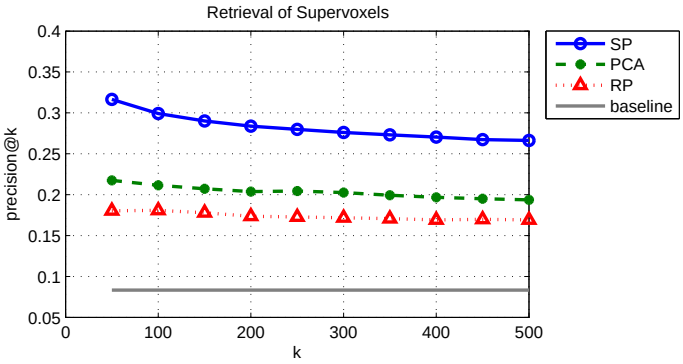


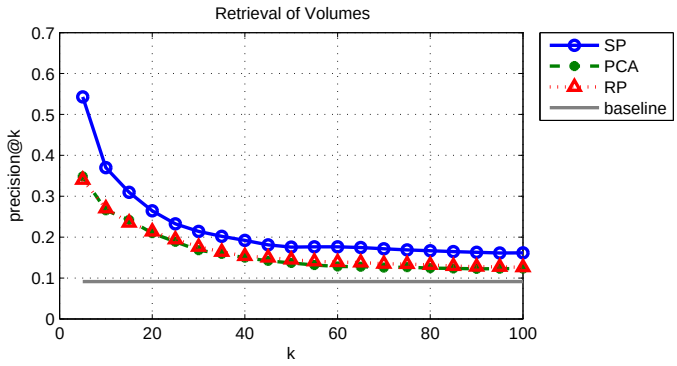


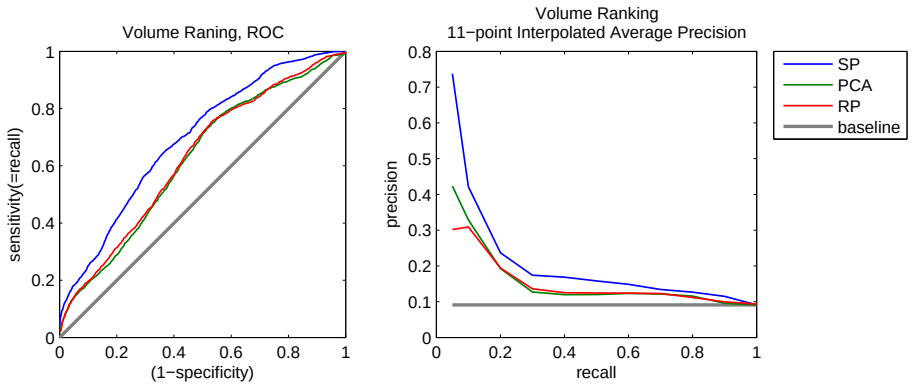


Quantitative retrieval results for Ground−glass

Anomaly: Ground−glass
Number of Queries: 63
Number of Supervoxels: 2220071
Number of Volumes (incl. query volume): 318

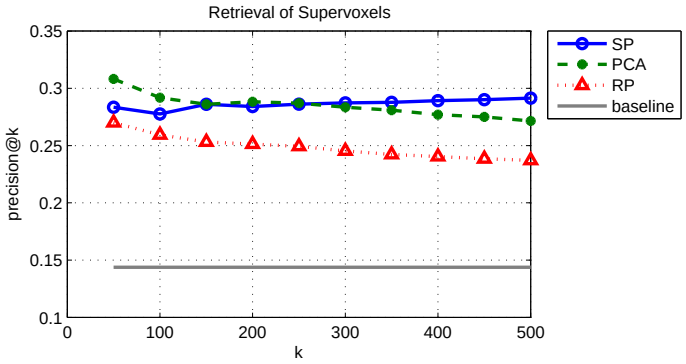


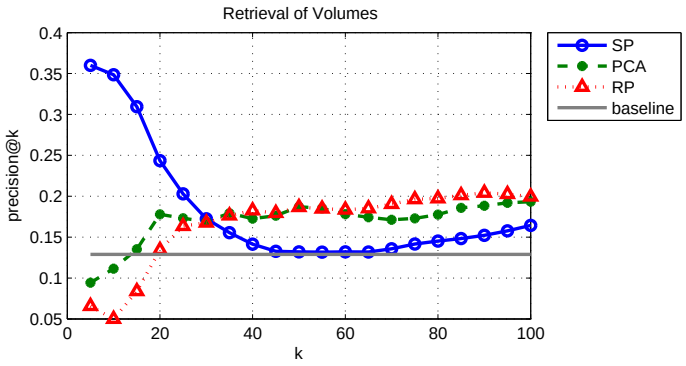


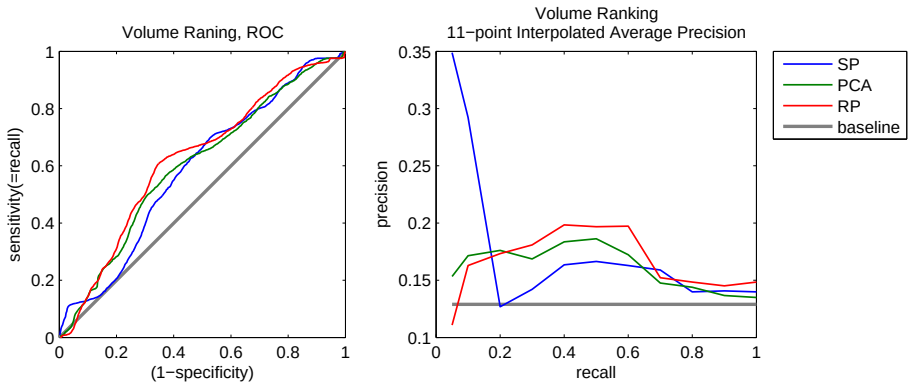


Quantitative retrieval results for Bulla

Anomaly: Bulla
Number of Queries: 34
Number of Supervoxels: 2220071
Number of Volumes (incl. query volume): 318

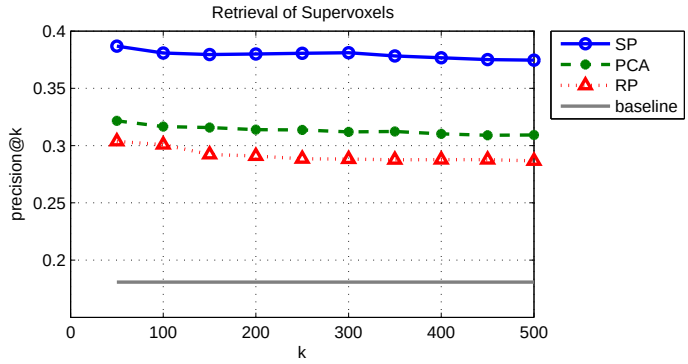


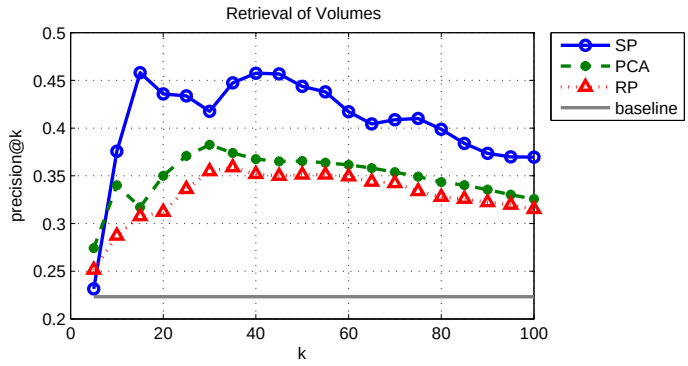


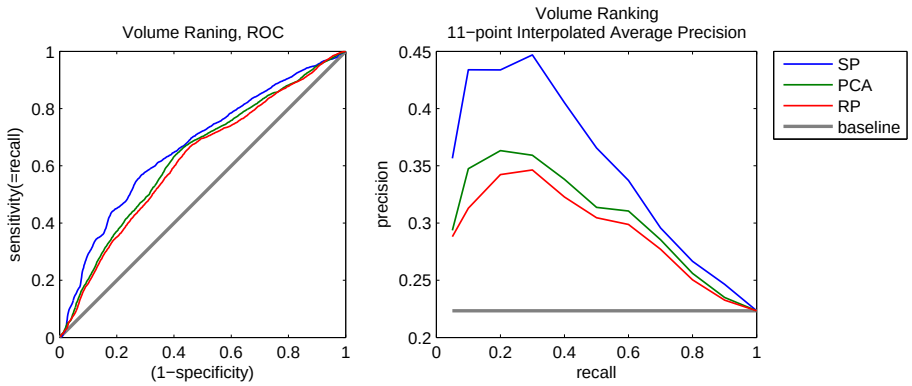


Quantitative retrieval results for Atelectasis

Anomaly: Atelectasis
Number of Queries: 39
Number of Supervoxels: 2220071
Number of Volumes (incl. query volume): 318







A.3 Results on Index Comparison

A.3.1 Approximated Retrieval and Volume Ranking

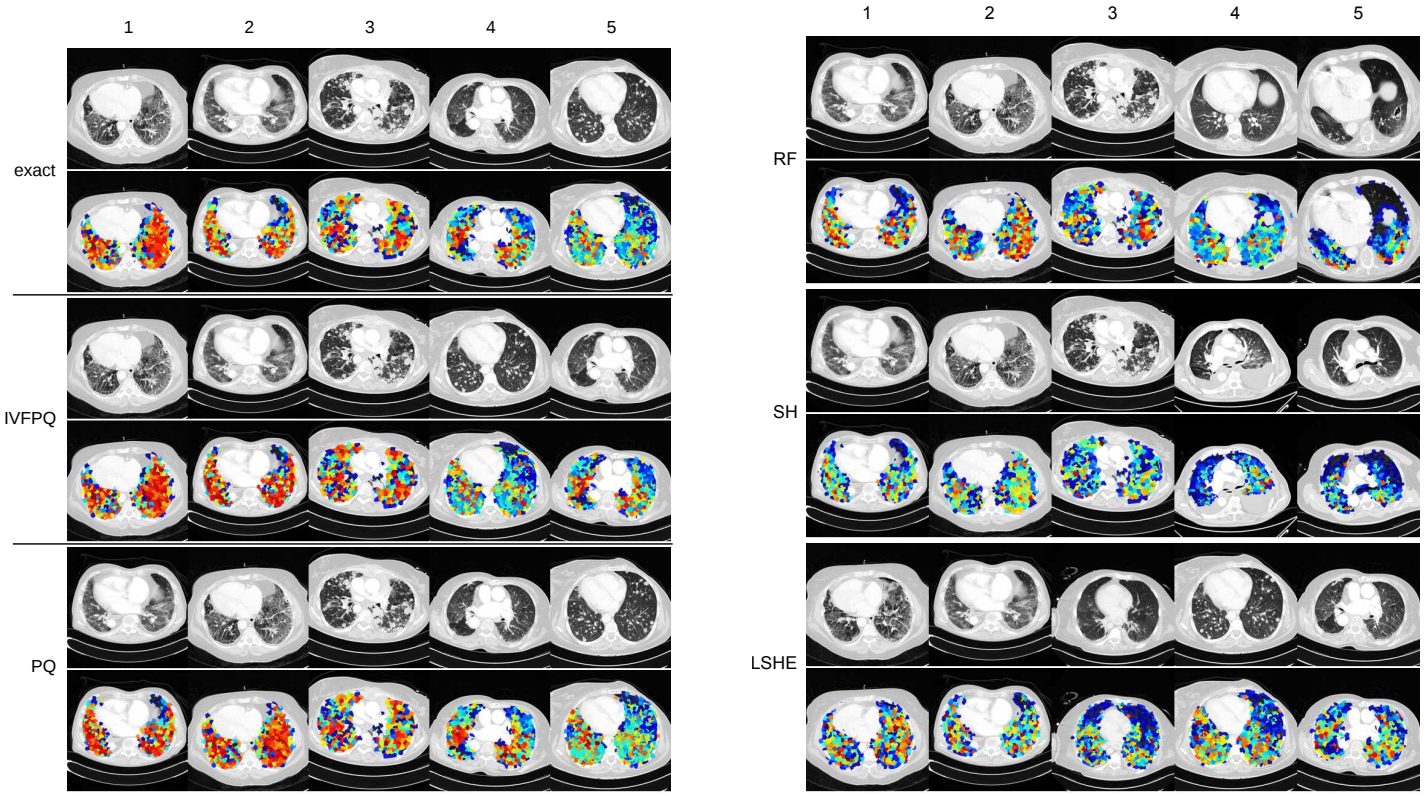


Figure A.6: Approximated volume ranking of rank 5 using exact euclidean distance, IVFPQ, PQ, RF, SH and LSHE. Results for 8 byte codes and ground-glass anomaly queries. The transversal slice with highest sum of supervoxel similarities is shown for each volume.